The Interactive Computer Graphics Book

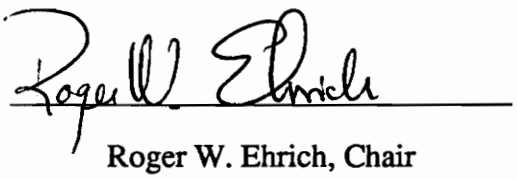
An Interactive Exploration Environment for Facilitating the Development of Visualization Skills

by

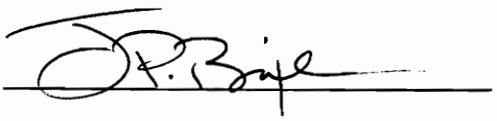
Lucia Inez Siochi

Project submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Science

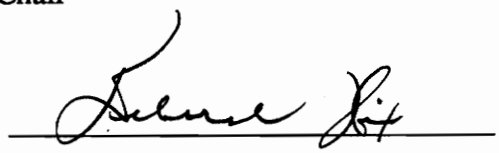
APPROVED:



Roger W. Ehrich, Chair



J. Patrick Bixler



Deborah Hix

April, 1991

Blacksburg, Virginia

An Interactive Exploration Environment for Facilitating the Development of Visualization Skills

by

Lucia Inez Siochi

Roger W. Ehrich, Chair
Computer Science

(ABSTRACT)

This report discusses an environment that may be able to facilitate the development of computer graphics students' visualization skills. The ability to visualize an image synthesized using a particular graphics technique is a skill needed in teaching and learning computer graphics. This skill has traditionally been developed inefficiently, using static, two-dimensional figures of synthesized images.

A method that may be more effective may be to allow the teacher or student to change the values of the parameters upon which the image is based, and to see the new image immediately. This approach to interacting with figures is integrated with textual material through a book metaphor.

The key aspect of this approach is that the student can explore and immediately see the effects of altering the values of various parameters.

Acknowledgements

I would like to thank Dr. Bixler for the idea of having an interactive book with dynamic figures as a tool to aid in Computer Graphics visualization. His support for the project and the many discussions in which the concept was developed and crystalized are greatly appreciated.

I would like to thank Dr. Ehrich and Dr. Hix for their helpful comments and for their valuable support of this project.

I would also like to thank the people who provided technical assistance. They made the task of learning C and how to use the Toolbox less burdensome.

Finally, I would like to thank my friends and family for their support through all the difficult times, especially my parents, whose generosity allowed me to pursue a good education.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Computer graphics is a highly visual field that is directly concerned with the production, use, manipulation, and dynamics of "electronic pictures." Thus, the communication of such material involves the presentation of visual information and its dynamics.

At Virginia Polytechnic Institute and State University (Virginia Tech), as well as at many other schools, the presentation of computer graphics material is attempted through the media of chalk and chalkboard, pencil and paper, textbook(s), and the instructor. The presentation of material includes the production, examination, and manipulation of two-dimensional representations of figures, objects, diagrams, and other visual models, and the exploration of the dynamics of these activities.

As with many other manually executed activities, this one could easily be seen as cumbersome, (and has been seen as such by both faculty and students), and could be done more efficiently and effectively if automated properly. Some faculty and students at Virginia Tech have recognized a "perceived" or "felt" need, (as opposed to one determined empirically), for the proper automation of such an activity [BIXJ88].

The automation should be an interactive environment allowing exploration. For accessibility, it should run on a system that is available for use by the teachers and students of the computer graphics classes at Virginia Tech. Preferably, it would be inexpensive and require minimal additional software and hardware.

This project begins the development of and lays the groundwork for such an automation. It is a software package (including documentation) that adds dynamics to the visual

interpretation of information. The computer program, however, does not provide any assessment or evaluation of student performance; it is interactive, but it is limited to presenting material and providing an environment for exploratory learning.

Since the scope of such an endeavor is very broad, this project is limited to designing and implementing a sample section and many of the underlying functions of the package. This initial implementation is designed and implemented so that the package will be usable and expandable.

BACKGROUND

There are two kinds of graphics packages. There are the application graphics packages designed so that the user can create graphics without knowing how the program does it. It does not require any programming from the user. Examples include paint or draw programs such as Canvas 2.1, Computer Assisted Design (CAD) programs such as ClarisCAD 2.0, and business or scientific graphing programs such as Cricket Graph.

Other graphics packages are extensions to programming languages, and include routines and functions for displaying and manipulating graphics output. With these packages, to create graphics, one must do some programming, [HEAD86, p. 50]. Examples include CORE, GKS, and PHIGS.

Typically, a programmer uses transformations of points from one coordinate system to another. The basic set of transformations and the order in which they are performed are shown in the viewing pipeline in figure 2-1.

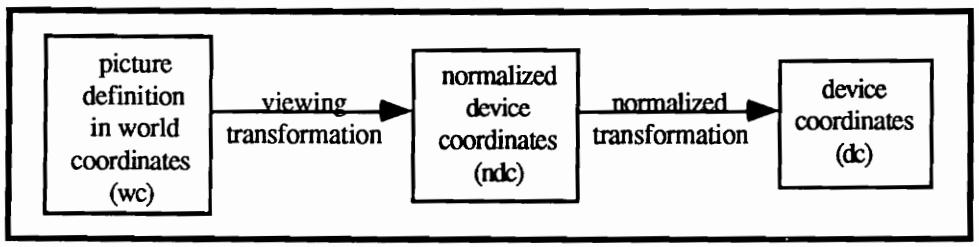


Figure 2-1. A simple viewing pipeline showing coordinate systems and transformations.

The coordinate systems used are often Cartesian and may be left-handed or right-handed. A scene in a coordinate system is a set of objects belonging to that coordinate system. Objects belonging to the *x* coordinate system are called *x objects* or *xo's*, e.g., an object in a world coordinate system is a world object or wo.

The coordinates referenced by the programmer (or user) are called *world coordinates (wc);* these are used to describe a scene, and are often set up to be convenient for use with the application domain. *Device coordinates (dc)* are those used by a particular output device, such as a monitor or a plotter. *Normalized device coordinates (ndc)* are used to represent the "processed" world coordinates before they are transformed into device coordinates; this is done in a way that facilitates conversion to different devices.

The graphics package projects a given view of a world coordinate picture definition on an output device. With the viewing transformations, the programmer can specify the view to be presented and the part of the output display to be used. [HEAD86, pp. 50-51]

Figure 2-2 is a more detailed viewing pipeline that shows the coordinate systems and transformations used to obtain the projection of a three dimensional world object onto a plane representing the screen or some other two dimensional device.

Viewing coordinates (vc) are used to represent the point of view of a viewer of the world scene. A scene can be viewed in a variety of ways, and the viewing coordinate system is used to describe how the scene is viewed.

The *viewing transformation* is a transformation of coordinate axes, and is used to obtain the description of an object in another coordinate system. A *projection transformation* is used to obtain the projected image of an object.

Since an object belonging to a certain coordinate system can be described in other coordinate systems, a subscript is added to indicate which system is being used to describe the object. For example, wo_{wc} refers to a world object wo being described in world coordinates, and wo_{vc} refers to a world object wo being described in viewing coordinates.

To perform a projection of a world object wo_{wc}, first obtain the description wo_{vc} of the object in viewing coordinates with the viewing transformation, then project wo_{vc} onto the viewing coordinate systems' z=0 plane with the projection transformation.

To show what the viewer sees, the projection of the world object is normalized and then mapped to a device's coordinates, such as the screen's coordinates.

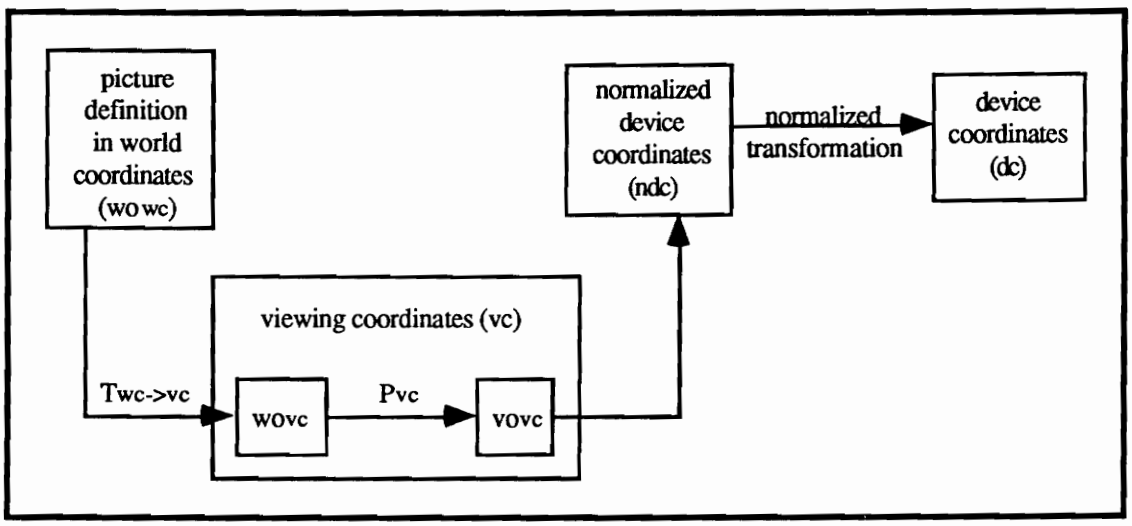


Figure 2-2. A viewing pipeline for three dimensional parallel projections.

THE INTERACTIVE COMPUTER GRAPHICS BOOK

The Interactive Computer Graphics Book (ICGB) was designed and partially developed to address the issues mentioned in the Introduction. This section describes the package, its interface, and some of the implementation.

The Package

This subsection describes the package. It describes where the package will work, the major parts of the final package, what each one of these parts is, how the parts relate to each other, and what has already been implemented.

A. System Configuration

The program runs on the Macintosh II under the A/UX 2.0 operating system, (for accessibility, since the Graphics classes are taught using such a configuration). It is written in the C programming language and uses the routines of the Macintosh Toolbox ROM (later referred to as the Toolbox).

B. Package Overview

The package has two major parts, namely, the program and the support materials.

The program will model a book, whose diagrams, illustrations, and visual models will be dynamic and can be modified interactively. When it is completed, it will provide a title page, a table of contents, sections, a glossary, and an index. Currently, it is limited to the

table of contents, containing a partial list of possible sections, a partially implemented sample section on Three Dimensional Oblique Parallel Projection, and a glossary containing sample entries.

The support materials will be the User's Guide and the Developer's Guide, which are subject to and address the limitations of the initial implementation.

C. The Program

The title page is a screen or window containing information about the program, such as the program name, author name, date of creation, version number, etc. It should appear when the program runs, i.e., when the book is opened, and should be accessible at any time while the book is open. This may be implemented as the "About program-name" menu item under the Apple menu, i.e., the menu, where program-name is replaced by the name of the program.

The table of contents screen displays a list of available sections and provides a way to choose and hence enter one of the sections. Each section has an "abstract" associated with it that briefly explains what is covered in the section, and what the prerequisites and the objectives of the section are.

The table of contents is directly accessible from any section, and the abstracts, sections, glossary, and index should be directly accessible from here. The user can initialize the state(s) of any or all sections, and can terminate the program from here.

The glossary provides access to the definitions of key words and concepts, and should be directly accessible from the table of contents and from any section.

The index contains a list of key words, concepts, and phrases, and the respective sections in which they are found or addressed. It should be directly accessible from the table of contents and from any section.

There are two kinds of sections, namely the introductory section, and the topic sections. The former is an introduction to the program; it discusses how to interact with the program to get it to do what one wants it to do ; it explains how to use the program, how to move around in the program, and how to move around and to explore in a section.

A topic section allows the presentation of material on the topic for that section, and provides an environment for exploratory learning of the material presented. There are several things a user can do in any topic section:

- read the abstract,
- study,
- find out how to explore,
- explore,
- request an elaboration,
- access the title page,
- access the table of contents (and from there, possibly go to another section or read section abstracts),
- access the glossary,
- access the index,
- insert or remove a bookmark in the section,

- remove all bookmarks from the book, (i.e., all sections),
- "open" the book or a section at a bookmark, and
- "close the book," (i.e., quit the application).

Studying provides a review of concepts needed, and a presentation and discussion of concepts being covered, including examples. This is done with text and with accompanying illustrative graphical activity.

The how to explore feature describes and illustrates the skills needed to be able to explore in that section, as well as how to use those skills.

Exploring allows the user to learn by discovery. The user should be able to manipulate variables, matrices, etc., and see the effects of the manipulation. An algorithm being covered may be available for tracing of statement execution at variable speeds.

A request for elaboration of a key concept, word, or phrase should result in further explanations of the selected item, if any such elaboration is available. The user must have available a way to find out what is available for elaboration, a way to communicate the desire for elaboration, and a way to specify what item is to be elaborated.

A bookmark holds one's place in a section. Inserting a bookmark tells the program that a user wants to be able to return to that same spot in the section under the same conditions that are present when the bookmark is inserted. Removing a bookmark tells the system that the user no longer cares to readily come back to that spot under those conditions. One can open the book to a section "at any given bookmark." One can also remove all bookmarks at once.

A section may have any of the following components: text areas, graphical activity areas, variables areas, matrix windows, algorithm windows, and menu areas. Text areas must be present in all sections. There should be a specific area where study text appears. Other kinds of text, such as dialog, elaboration, etc. appear in their own windows or dialog boxes.

Graphical activity areas must also be present in all sections. These are areas where graphical activity occurs. These areas may be manifested in the form of one or more windows, depending on how many may be necessary.

Variables areas are areas where one finds the values of variables, and where one changes these values. They may appear as one or more windows.

Matrix windows display matrices and allow changes to be made to the values of the matrix's components.

Algorithm windows display algorithms. When an algorithm is being stepped through or traced, the currently executing statement should be identifiable. Trace speed may be defined as a menu item.

Menu areas must be present in all sections. One area that must be present is the menu bar at the top of the screen. The abstract, table of contents, glossary, index, bookmarking activities, and closing the book must be accessible from the menu bar. Other menu areas may be used if deemed necessary or useful, but care must be taken to maintain consistency across command names, forms, and behaviors.

D. The Support Materials

The support materials are documents that provide information on the proper use and growth of the program. The User's Guide, [SIOL91b], explains how to use the application. It includes directions for starting, for stopping, and for moving around among and for working within the different parts of the running program.

Information related to the technical aspects of the implementation and development of the program is contained in [SIOL91a], the Developer's Guide. It includes directions for adding new sections.

E. Implemented Items

Only some of the items described in the subsection The Package are available. Below is a list of the items that are implemented and the degrees to which they are implemented.

- Table of Contents. This allows the user to select an implemented section. Some sections that are not implemented are also listed. The glossary is available from here.

- Sections. One section, ("3D Oblique Parallel Projections"), is partially implemented. Text is available for reading, and keyword elaborations are accessible. Figures are accessible and they can be modified through the available parameters. Some options related to the figure are available. The user can return to the Table of Contents from here.

- Glossary. From a section, one will be able to access the Glossary entry of some key words or phrases. Currently, the implementation of this feature is not yet completed.

- Menu Bar and Menus. The menu bar and some of the menu items are implemented. See the Interface subsection for the list of menus, menu items, and for information on which menu items' functions are available.
- The Support Materials. Both the Developer's Guide and the User's Guide are written and are included as an appendix.

Interface

The interface is a direct-manipulation interface, similar to a typical Macintosh interface, which makes use of windows, pull-down menus, and the mouse. This interface has several parts, namely, the Table of Contents, a section and its parts, the Glossary, and the Menu Bar.

The Menu Bar has two pull-down menus with the Table of Contents, namely, the Menu and the File Menu. Two more are available with a Section, the Window Menu, and the Options Menu. Below is a list of the menu items and a description of each item.

 Menu

- About The ICGB... - This is the title page. It is not implemented.
- Help - A help facility. It is not implemented.

File Menu

- Table of Contents - The user leaves the current section and enters the Table of Contents. This is available only from the current section.
- Resume Section - The user leaves the Table of Contents and returns to the current section. This is available only from the Table of Contents.

- Close Section - The current section is closed and the user returns to the Table of Contents if the user is not already there. This is available from both the Table of Contents and the current section.

- Glossary - The glossary entries appear. This will be available only from the current section.

- Quit - The program stops execution. This is available from both the Table of Contents and the current section.

Window Menu

- Text - This toggles the Text window's visibility. A check mark is present if the window is visible and is absent if the window is not visible.

- Figure- This toggles the Figure window's visibility. A check mark is present if the window is visible and is absent if the window is not visible.

- Parameters- This toggles the Parameters window's visibility. A check mark is present if the window is visible and is absent if the window is not visible.

The menu items of the Options menu may differ among the sections. Each section would have different options appropriate for that section. The menu items listed below are available in the "3D Oblique Parallel Projection" section.

Options Menu

- Projector Lines - This toggles the visibility of the projector lines in the Figure window. A check mark is present if the projector lines are visible and is absent if the lines are not visible.

- World Axes - This toggles the visibility of the world coordinate axes in the Figure window. A check mark is present if the world axes are visible and is absent if the lines are not visible. This item is not implemented.

- View Axes - This toggles the visibility of the view coordinate axes in the Figure window. A check mark is present if the view axes are visible and is absent if the lines are not visible. This item is not implemented.

The Table of Contents is listed in a window that has no scroll bar and cannot grow. It can be moved to a different location on the screen. It contains text, each line of which refers to a different Section. When the user double-clicks on a Section, if that Section is implemented, the Table of Contents window will be hidden and the appropriate Section appears and becomes the "current section."

A Section has three main parts, namely, the Text, the Figures, and the Parameters. Each one of these parts appears in its own window. Each window can be made hidden or visible, and can be moved to a different location on the screen.

The Text window contains text, is resizable, and has a scroll bar. When the user double-clicks on a phrase of the form "figure n," where n refers to a figure number, then the nth figure in the section appears in the Figure window and becomes the "current figure." When the user double clicks on any other word or phrase, if the word or phrase has an entry in the Glossary, then the Glossary displays that entry.

The Figures window displays the current figure, and is resizable. Features that affect the appearance of all figures in the Section are available through the Options Menu in the Menu Bar.

The Parameters window, which is actually a dialog box, contains parameter labels and their respective values, an Apply button, and a Reset button. Labels are not editable, but the values are. When the user clicks with the mouse in the Apply button, the parameter values currently displayed are then applied to the figure. When the user clicks in the Reset button,

the parameter values are set to the default or original values, and the figure is modified accordingly.

The Glossary dialog box contains an entry and a Done button. Each entry is composed of text and has an entry name and a description. When the user clicks in the Done button, the dialog box is dismissed.

Implementation

A. Structure and Organization of the Code

The program is designed to facilitate some modifications and additions. It is composed of the modules listed below, and uses various input files and include files.

- event module - This is the main program; it handles events and the event loop.
- menu module - This takes care of most menu related data structures and routines.
- window module - This takes care of most window related data structures and functions, and contains routines that manage the user's traversal within the "book." This module also takes care of defining and manipulating the contents of the text window.
- figure module - This takes care of most figure and parameters related data structures and routines. Note that this module currently handles only figures for three-dimensional wireframe views and parallel projections.
- matrix module - This contains matrix and vector operations for the figure module.
- utilities module - This contains various routines useful for debugging.
- resource file - This is the resource file; it contains the resources that the program uses.

- input files - These files include information about the Table of Contents' text, and about a section's text, figures, and parameters.

[SIOL91a] contains details on making additions to the ICGB.

B. Addition of the Missing Parts of the Viewing Pipeline

The Macintosh Toolbox ROM is a graphics package extension to Pascal, C, or Assembly Language. Although it is a programmer's package, it does not directly support some parts of the viewing pipeline that the other packages normally support. It is oriented more towards interface generation and manipulation rather than towards picture creation, manipulation, projection and transformation. It does have picture creation and manipulation capabilities, but it lacks much of the projection and transformation capabilities. Since it supports only device coordinates, it is missing most of the viewing pipeline shown in figure 2-1.

The Interactive Computer Graphics Book's program includes routines to take care of some of the missing parts. These are included in the figure and matrix modules.

C. Modifications to the 3D Parallel Projection Viewing Pipeline

The parallel projection viewing pipeline in figure 2-2 is not sufficient for the projections needed by the ICGB. The pipeline produces a two dimensional projection of a world object, whereas the ICGB needs two dimensional projections of both a world object *and its projection.* To support this, a generalization of the projection procedure was obtained and used, and the metaview coordinate system was added, producing the metaviewing pipeline as shown in figure 3-1.

A generalization of the Background's procedure for performing a projection is as follows: An object so belonging to a source coordinate system srcc is projected onto the z=0 plane of the destination coordinate system dstc in two steps. Obtain the description so_{dstc} with a transformation of coordinate axes, then project so_{dstc} using the destination's projection transformation. The projection of the source object is said to be a destination object, e.g., the projection of a world object onto the viewing coordinate system is a view(ing) object or vo.

Metaview coordinates (mvc) are used to describe the point of view of the metaviewer. The metaviewer sees a metascene, which is composed of a world scene and a view scene; one sees both the world objects and the view objects from one's own point of view.

To show what the metaviewer sees, the world objects and the view objects are projected onto the metaview's z=0 plane. This projection is then normalized and then mapped to device coordinates, which, in this case, is the Figure window's coordinates.

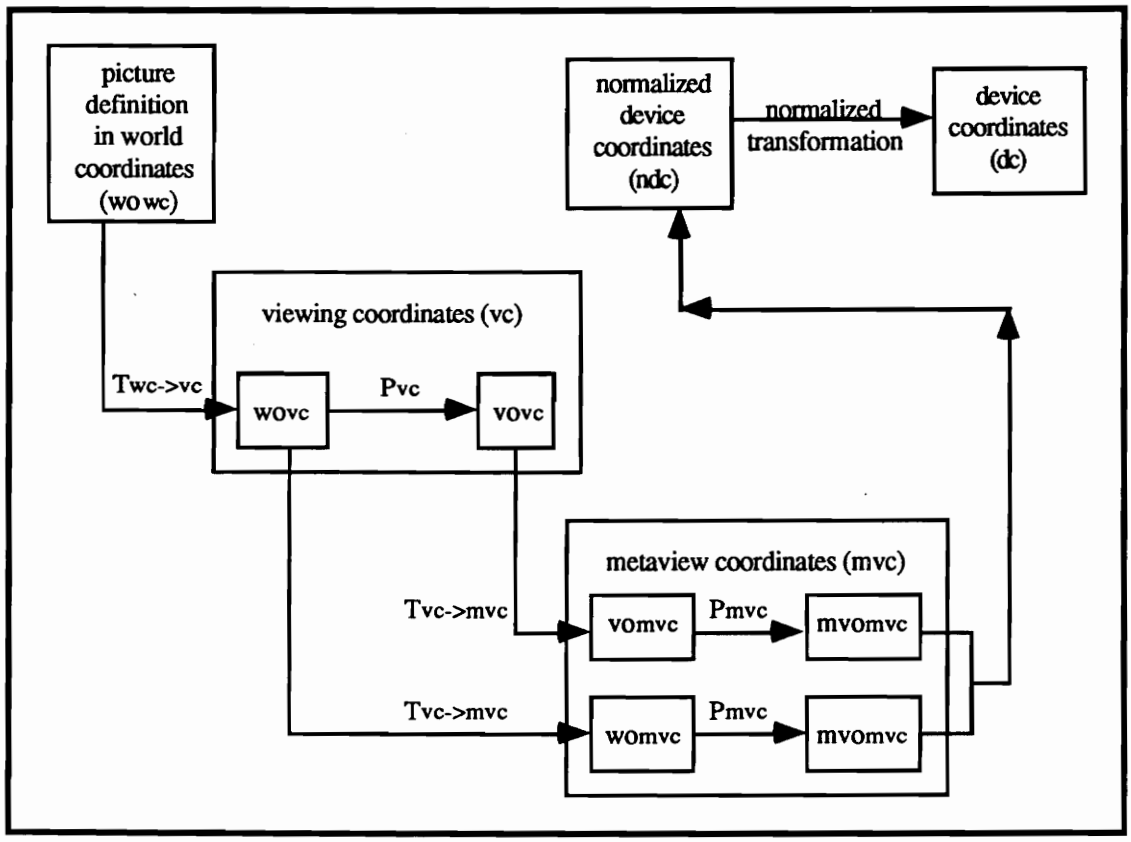


Figure 3-1. The metaviewing pipeline; a modification of the viewing pipeline for three dimensional parallel projection.

DISCUSSION

Factors mentioned in the Introduction affected the decisions made about the ICGB package. Decisions were made concerning the choice of computer, operating system, programming language and/or graphics package, and what topic to use for a section. Because of these decisions, certain challenges were encountered during the course of the package's development. Although development has ended, there still are many changes or additions that can be made.

Decisions

The package's intended users are primarily Computer Graphics students and instructors at Virginia Tech, so it was to be made available for use with the Computer Graphics class(es) at Virginia Tech. Since these classes are taught using the Macintosh II running the A/UX operating system, the package should also be available for use on the Macintosh II under A/UX for accessibility.

The Macintosh Toolbox ROM graphics package and the C programming language were chosen instead of a user-oriented application graphics package. This imposes no additional hardware or software expense on either the users or the developers. This also allows the developer greater freedom and flexibility in customizing the package than an application graphics package would. Also, technical support in the forms of documentation, such as [APPL88a], [APPL88b], [APPL90], [CHES90], [KERB88], [TAKJ86], [MATK88], and of people who were knowledgeable in C and in the Toolbox, were available.

Other graphics packages and languages were considered. Pascal was considered as an alternative to C. However, there was no free Pascal compiler available at the time that seemed reliable enough. The X Window System (X) was also considered, and would have been a good alternative because of the portability it would give to the package. However, at the time that development began, documentation on X was very scarce, and X itself was not available on A/UX. Software packages such as Mathematica would impose an additional expense on the users and the developers, and would limit the developer's ability to customize the package.

HyperCard was considered more carefully, because it was relatively inexpensive and facilitated the development of applications that needed some hypertext abilities through its linking facility. A prototype was developed to examine the feasibility of developing the ICGB on HyperCard. A formative evaluation of the prototype revealed that the Book metaphor was considered useful [SIOL89]. However, implementing the prototype revealed several problems related to the feasibility of using HyperCard for the final implementation. These problems are listed below.

- The card size was too small to be able to fit the necessary information on it. HyperCard's card size remains the same regardless of the screen size. This was done for backward compatibility with the Macintoshes with smaller screens.

- Only one card can be visible at a time. Being able to have multiple cards would have helped get around the problem of the small card size, because the information can be distributed among the cards. It was considered important to be able to have the text, figures, and parameters visible all at the same time.

- The prototype's data on the cards were hard-wired. It would be preferable to have a more flexible way of obtaining the data.

- HyperCard has limited mathematical capabilities. To perform the necessary calculations, code would have to be written in a programming language and linked to HyperCard through what are known as XCMD's and XFCN's.

- HyperCard was not available on A/UX. Insistence on using A/UX precluded the use of HyperCard at the time.

Some time after these decisions were made and the implementation was well under way, some of these restrictions and problems no longer applied. For example, since A/UX 2.0 was released, users have access to both the Macintosh OS partition and the A/UX partition, at the same time, through MultiFinder. Also, the X Window System version for A/UX was finally released and documentation became more plentiful. By that time, however, the implementation had progressed far enough that changing the basic environment was not considered feasible.

The topic "3D Oblique Parallel Projection" was chosen for two reasons. First, it was felt that using this topic would be a good demonstration of the possible usefulness of the package. Second, it is a subtopic of a topic which, according to [ANDB88], is covered in many Computer Graphics classes across the United States, and in [HEAD86] and [FOLJ90], which, according to the same article, are commonly used Computer Graphics textbooks.

Challenges

Several challenges were encountered during the development of the ICGB. They are listed below.

- The C programming language and the Macintosh Toolbox ROM had to be learned. Since the implementation was to be in C using the Toolbox, the author/programmer

had to learn what was necessary to implement the ICGB. Because both C and the Toolbox are quite complex, this was not a trivial task. It was also difficult to determine whether programming errors were due to the improper use of C or of the Toolbox.

- Information about the use of the Toolbox with A/UX was scarce and difficult to understand. [APPL90] was the only document available to the author that covered the differences in the use of the Toolbox on A/UX and on MacOS. Because most Toolbox programmers use the Toolbox only on the MacOS, and most of the Toolbox references are written from the MacOS perspective, information on the differences was difficult to obtain, and sometimes difficult to understand.

- The ICGB had to be ported from A/UX 1.1 to A/UX 2.0. A large portion of the implementation of the ICGB was done under A/UX 1.1. When A/UX 2.0 was released, it was installed as an upgrade on the Macintosh II's available for ICGB's development and for the students' use in the Computer Science Laboratory. In order for development to continue, the ICGB had to be ported to A/UX 2.0.

- The ICGB had to be MultiFinder compatible. Toolbox applications that run on A/UX 2.0 must be MultiFinder compatible, because A/UX 2.0 uses MultiFinder to allow the user to move between A/UX and MacOS. Applications do not have to take advantage of capabilities available to them from MultiFinder; they only have to be well-behaved and stable in the MultiFinder environment.

- The ICGB needed to accommodate future modifications and additions. It is written in a modular fashion, and facilitates the modification of already existing sections, and, to some extent, the addition of other sections and their components, as discussed in [SIOL91a].

- Various data structures containing information about figures had to be defined. What information needed to be stored and what form would facilitate its use were determined.

- Some parts of the viewing pipeline were implemented. The Macintosh Toolbox ROM does not support some parts of the viewing pipeline. The details are discussed in the second section's Implementation subsection.

- The parallel projection pipeline was modified, and the modifications were implemented. The pipeline was missing necessary operations, as discussed in the second section's Implementation subsection.

Future Work

Much work remains to be done; additions, modifications, and extensions can be made to the package, and formative and summative evaluations of the package need to be made.

Some of the features mentioned in the third section's subsection The Package are not implemented. They are the abstracts, bookmarks, the introductory section, matrix windows, algorithm windows, and the index.

Modifications can be made to the program. These include modifying the pipeline in figure 3-1, and completing partially implemented items.

The metaviewing pipeline shown in figure 3-1 needs modification. Informal testing with users has revealed a behavior of the figure that needs to be changed. The world object moves to a different place in the Figure window when the value of a parameter defined in view coordinates is changed. The resulting projection is mathematically correct, but it would be more useful to the user if the world object remained in the same position relative to the window. This gives the user a reference point from which to make comparisons with other instances of the figure. Apparently, the figure behaves this way because the world object is first converted to view coordinates before being converted to metaview coordinates. The pipeline should be modified so that the world object is converted directly to metaview coordinates.

Some program features are only partially implemented. The menu items present in the program, but whose functions are not implemented, are listed below.

- Help
- About The ICGB...
- World Axes
- View Axes

Descriptions of the functions of these menu items are given in the Interface subsection of the third section.

The Table of Contents does not have a scroll bar. Eventually, it will need one when the text has more lines than those that will fit in the window.

Key words in the Section text need some form of highlighting, and an option to toggle the highlighting on and off. This option may be made available through the Options menu.

The program is not robust. Some currently existing features need to be fixed. Their behavior needs to be more well-defined; error correction has not been implemented, although some error detection and avoidance have been.

- Check if the compiled resource file is present in the same directory as the program. If absent, put up a dialog box telling the user that the said file needs to be present in the same directory as the program, and terminate the program when the user dismisses the dialog.
- Check if input files are present in the same directory as the program and if they are all in the correct format. If a file is absent, or if there is a mistake in the format, put up a dialog box telling the user that the said file(s) need to be present in the same

directory as the program, or what the correct format is, as appropriate. Terminate the program when the user dismisses the dialog.

- Check the format of any input parameter values. If there is a mistake in the format, put up a dialog box telling the user that the format is incorrect and what the expected format is. After the user dismisses the dialog box, reset the parameter value to its previous correctly formatted value.

- Define and implement the behavior of the program if memory was requested but not obtained. The program already checks for a failed memory request, and does not run code that would have used that memory. However, it should also put up a dialog box informing the user of the problem.

Extensions can be made to the program to make it more flexible.

- More sections can be added to cover other topics related to three dimensional viewing and projection. Minimal additions may be needed, such as one or two more routines or fields in the figure structure. The input files will also need additional data describing the new sections.

- Currently, it can handle only sections on three dimensional viewing and parallel projections. Data structures and routines need to be added to support other types of sections. Perhaps a variant record structure can be used for the figure data structure. Research can also be done on determining what is a general form, or what are types of forms, of the figure data structure that will cover most, if not all, the topics in Computer Graphics.

- The formats used by the input files are quite rigid. Being able to use blank lines, which currently is not supported, would improve readability, making the formats more flexible.

The package has not been evaluated extensively. Some informal testing was performed by the programmer and by some users, which revealed some of the items mentioned above. Other evaluation procedures, which are listed below, may still be performed.

- Formative and summative evaluation of the interface. The current interface is the first implementation and needs to be refined through testing with users. Interface testing and refinement was done on the HyperCard prototype, but not on the Toolbox implementation.

- Formative and summative evaluation of the support materials. Similarly, these documents are initial versions that are yet to be refined through testing with users of the documents.

- Evaluation of the package's effectiveness as a tool to aid the teaching and the learning of Computer Graphics material. A rigorous and conclusive evaluation may be difficult, if not impossible, to perform. However, what would be sufficient is a more informal testing that shows evidence that seems to support or refute the hypothesis that the package is effective as a teaching and learning tool. This may also reveal how the package can be used more effectively.

Any evaluation should be performed at two times: when the item being evaluated is still a novelty, and when some time has passed and it is no longer so novel. This can help show what results were due to a "halo effect" of the novelty of the item, and what were not.

CONCLUSION

There is a perceived need for automated generation of figures used in the presentation and clarification of computer graphics material. In an effort to answer this need, a software package was partially designed and implemented to provide an interactive environment that partially automated the generation of such figures. Automatic figure generation in an interactive environment can allow the user to explore and can help the teacher to clarify the subject matter.

Decisions were made based on various options and on conditions that were necessary or desirable. Several challenges, some of which were the result of the decisions made, had to be addressed.

The work that is yet to be done is outlined in the previous section. In spite of this remaining work, much of the foundation has already been laid. A sufficient amount of work has already been done so that, in a limited way, the package may already be used as intended. Hopefully, this and any future versions will prove to be useful in facilitating the development of computer graphics visualization skills.

REFERENCES

[ANDB88] Anderson, Bennett W., and Burton, Robert P. "Computer Graphics Curricula: A Survey of Ph.D. Granting Departments," Computer Graphics, Vol. 22, No. 2, (April 1988), pp. 94-98.

[APPL88a] Apple Computer, Inc. Inside Macintosh, Volumes I-V. Addison-Wesley Publishing Company, Inc., 1985-1988.

[APPL88b] Apple Computer, Inc. Programmer's Guide to MultiFinder. Cupertino, CA: Apple Computer, Inc., 1988.

[APPL90] Apple Computer, Inc. A/UX Toolbox: Macintosh ROM Interface. Cupertino, CA: Apple Computer, Inc., 1990.

[BIXJ88] Bixler, J. Patrick. Currently with Recognition Research, Inc., Blacksburg, VA. Informal discussions, Fall 1988.

[CHES90] Chernicoff, Stephen. Macintosh Revealed, Volumes 1-4. Carmel, IN: Hayden Books, a division of Howard W. Sams & Company, 1985-1990.

[FOLJ90] Foley, James; Van Dam, Andries; Feiner, Steven; and Hughes, John. Computer Graphics Principles and Practice, Second Edition. Reading, MA: Addison-Wesley Publishing Co., 1990.

[HEAD86] Hearn, Donald, and Baker, M. Pauline. Computer Graphics. Englewood Cliffs, NJ: Prentice Hall, 1986.

[KERB88] Kernighan, Brian, and Ritchie, Dennis. The C Programming Language, Second Edition. Englewood Cliffs, NJ: Prentice Hall,1988.

[MATK88] Mathews, Keith, and Friedland, Jay. Encyclopedia Mac ROM, A Complete A-Z Reference for Macintosh Toolbox and Operating System Routines. New York, NY: Brady, 1988.

[SIOL89] Siochi, Lucia; Berkowitz, Jack; and Tan, Kay. "Graphics Tutorial Prototype Initial Evaluation Stage." In "Computer Graphics Tutorial Interface Prototyping Project: Executive Summary and Compilation of Semester Reports," class project for CS 5714: Human-Computer Interaction, Computer Science Department, Virginia Tech, Blacksburg, VA, Spring 1989.

[SIOL91a] Siochi, Lucia. Developer's Guide for The Interactive Computer Graphics Book. Computer Science Department, Virginia Tech, Blacksburg, VA, April 1991.

[SIOL91b] Siochi, Lucia. User's Guide for The Interactive Computer Graphics Book. Computer Science Department, Virginia Tech, Blacksburg, VA, April 1991.

[TAKJ86] Takatsuka, Jim; Huxman, Fred; and Burnard, David. Using the Macintosh Toolbox with C. Berkeley, CA: SYBEX, Inc., 1986.

APPENDIX: THE USER'S GUIDE AND THE DEVELOPER'S GUIDE

USER'S GUIDE

for the

INTERACTIVE COMPUTER GRAPHICS BOOK

by
Lucia I. Siochi

Computer Science Department
Virginia Tech
Blacksburg, VA 24061
siochil@csgrad.cs.vt.edu

April 1991

ABOUT THIS GUIDE

This is the User's Guide for the Interactive Computer Graphics Book (the ICGB, or the Book). The ICGB provides you with an environment suitable for exploratory learning in the area of Computer Graphics. This guide contains information that can help you with using the Book.

For advanced users, and for those interested in adding more sections to the Book, see the Developer's Guide for more details.

CONVENTIONS USED IN THIS GUIDE

This guide uses standard A/UX 2.0 and Macintosh OS terminologies.

Text in *italics* should be typed in as is. Text enclosed in <brackets> should be replaced by the appropriate text as described within the brackets. Key combinations may also be enclosed in brackets, for example, <ret> means "press the return key."

Words appearing in **this font** refer to items in the application, such as menu items, window labels, and button labels.

TOC means "Table of Contents."

WHAT YOU WILL NEED

This chapter lists the items that you need to be able to use the Book.

System Configuration

You will need the following configuration:

- a Macintosh II running
- A/UX 2.0 with
- the "mac32" environment set up, and
- a CommandShell window open.

Skills

You must have certain computer skills, called basic skills, to be able to use the Book. Other skills, called advanced skills, may be helpful.

Basic Skills

- ability to log on to A/UX 2.0
- ability to set up the mac32 environment
- ability to perform actions according to the Macintosh OS environment's user interface paradigm

Advanced Skill(s)

- familiarity with A/UX 2.0
- some understanding of the material of the Computer Graphics topic of the section you will be covering

Files

You will need certain files in your A/UX working directory to be able to use the Book. If they are not present, you will need your installation disk, and you must install the Book on your working directory on A/UX using this disk. See the chapter **INSTALLING THE BOOK** for directions on how to install the Book.

To use the Book, you will need these files:

- gt
- %gt
- toc.txt
- *.txt
- *.fig
- *.prm

where * is a wild card.

INSTALLING THE BOOK

If any of the files listed below are missing, you must install the Book. This is done by making sure that these files are in your working directory on A/UX. Copies of these files are available on your installation disk. Simply copy these files from the installation disk to your A/UX working directory.

These are the files that you will need to have in your A/UX working directory:

- gt
- %gt
- toc.txt
- *.txt
- *.fig
- *.prm

where * is a wild card.

ABOUT THE BOOK

As its name implies, this application is similar to a book. It has a Table of Contents, one or more Sections, and a Glossary. Each section has text and accompanying figures related to a particular topic.

However, the ICGB is more than just a static book. It provides you with an environment suitable for exploratory learning in the area of Computer Graphics. It automates the production and manipulation of figures that are used to present and to clarify Computer Graphics subject matter.

This Book has the following parts:

- Table of Contents
- Section(s)
- Glossary
- Menu Bar

Each Section has the following parts:

- Text
- Figures
- Parameters

Although the Glossary is documented, currently, its implementation is not finished, so the Glossary features and capabilities are not available.

USING THE BOOK

This section tells you how to use the Book.

Getting Started

Before using the Book, make sure that it is installed in your working directory in A/UX. If it is not, see the chapter **INSTALLING THE BOOK** for installation instructions.

Also, be sure that you have the proper configuration as described in the System Configuration section of the chapter **WHAT YOU WILL NEED.**

After the conditions mentioned above have been satisfied, you may proceed.

At the prompt in the Command-Shell window, type

launch gt <ret>

This will start the application and "open" the Book to the Table of Contents.

Navigating Within the Book

While you are using the Book, you will be in one of two places. You will be either in the Table of Contents or in a Section. You may move back and forth between these places.

While in a section, you may return to the TOC without closing the section you are in. A section that you have opened and have not yet closed is called the "current section". At most one section may be current at a time.

Wherever you are in the Book, you will always have a Menu Bar. See the **Menu Bar** section below to find out what you can do through the Menu Bar.

When You're Done

When you are finished with the Book and wish to stop the program, pick the **Quit** menu item in the **File** menu.

The program will terminate, and you will be returned to the CommandShell window.

Menu Bar

The Menu Bar has two pull-down menus with the **Table of Contents**, namely, the Menu and the **File** Menu.

Two more are available with a Section, the **Window** Menu, and the **Options** Menu. Below is a list of the menu items and a description of each item.

Menu

- **About The ICGB...** - This is the title page. It is not implemented.
- **Help** - A help facility. It is not implemented.

File Menu

- **Table of Contents** - You leave the current section and enter the Table of Contents. This is available only from the current section.
- **Resume Section** - You leave the Table of Contents and return to the current section. This is available only from the Table of Contents.
- **Close Section** - The current section is closed and you return to the Table of Contents if you are not already there. This is available from both the Table of Contents and the current section.
- **Glossary** - The glossary entries appear. This is currently not implemented.
- **Quit** - The application terminates. This is available from both the Table of Contents and the current section.

Window Menu

- **Text** - This toggles the Text window's visibility. A check mark is present if the window is visible and is absent if the window is not visible.
- **Figure**- This toggles the Figure window's visibility. A check mark is present if the window is visible and is absent if the window is not visible.
- **Parameters**- This toggles the Parameters window's visibility. A check mark is present if the window is visible and is absent if the window is not visible.

Options Menu

The menu items of the **Options** menu may differ among the sections. Each section would have different options appropriate for that section. The menu items listed below are available in the "3D Oblique Parallel Projection" section.

Options Menu

- **Projector Lines** - This toggles the visibility of the projector lines in the Figure window. A check mark is present if the projector lines are visible and is absent if the lines are not visible.

- **World Axes** - This toggles the visibility of the world coordinate axes in the Figure window. A check mark is present if the world axes are visible and is absent if the lines are not visible. This item is not implemented.
- **View Axes** - This toggles the visibility of the view coordinate axes in the Figure window. A check mark is present if the view axes are visible and is absent if the lines are not visible. This item is not implemented.

Table of Contents

The TOC is listed in a window as shown in figure 1. It contains text, each line of which refers to a different section in the Book.

You may open a section from here. To open a section, simply double-click on the line of text referring to the section that you want. If that section is available, and there is no current section, then the TOC window is hidden, the corresponding section appears, and the menu bar is modified appropriately. This section is now the current section. If there already is a current section, you must close it before opening a section.

You may close the current section and may perform other actions through the menu bar. See the section **Menu Bar** above for more details.

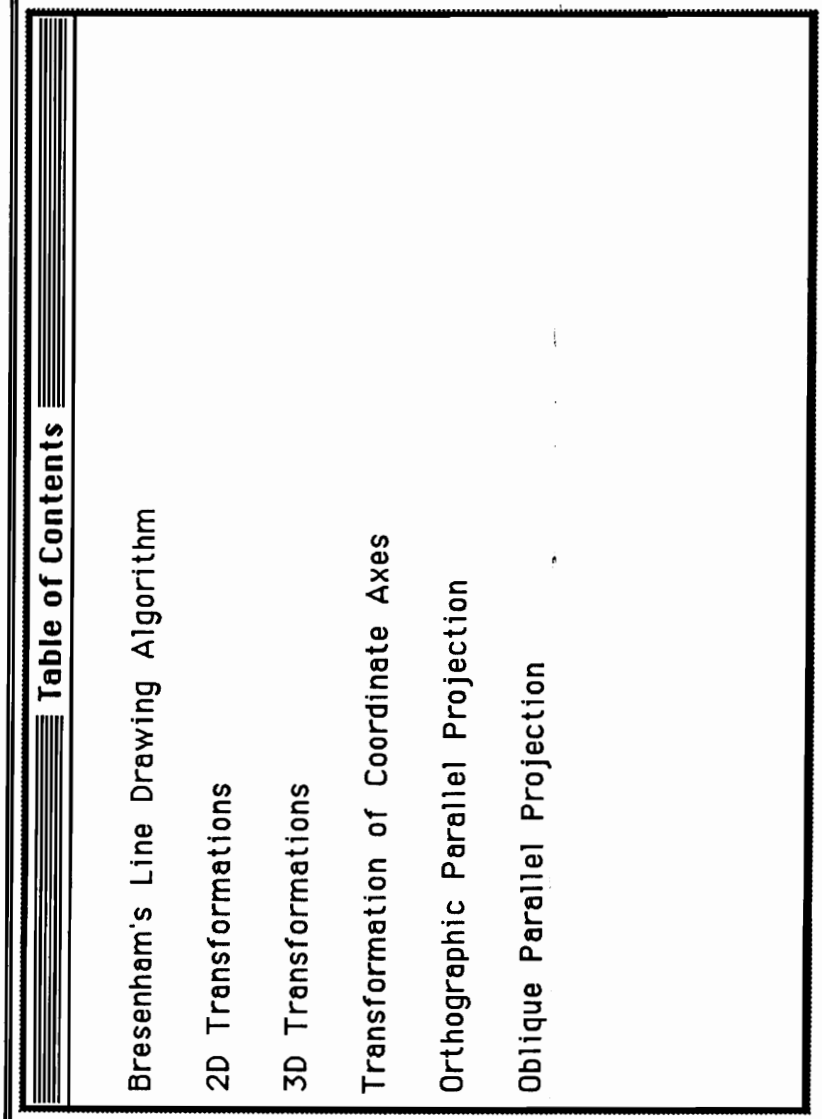


Figure 1. A side view of a screen dump showing an active Table of Contents window.

Section

A Section has three main parts, namely, the Text, the Figures, and the Parameters. Each one of these parts appears in its own window. Each window can be made hidden or visible, and can be moved to a different location on the screen.

Text

The **Text** window contains text, is resizable, and has a scroll bar. Figure 2 shows a screen dump of an active **Text** window.

When you double-click on a phrase of the form "figure n," where n refers to a figure number, the nth figure in the section appears in the **Figure** window and becomes the current figure.

When you double click on any other word or phrase, if the word or phrase has an entry in the **Glossary**, then the **Glossary** displays that entry.

Use the scroll bar to scroll up and down the text.

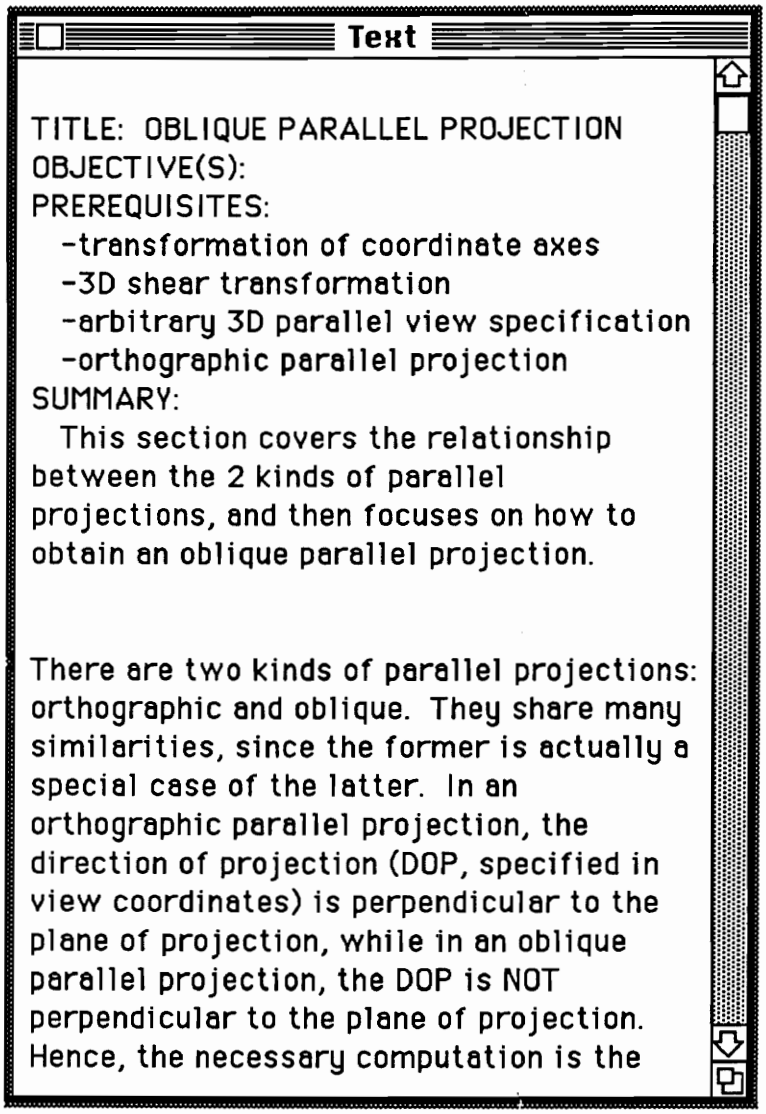


Figure 2. A screen dump showing an active Text window.

Figures

The **Figure** window displays the current figure, and is resizable. Figure 3 shows a screen dump of an inactive **Figure** window.

Features that affect the appearance of all figures in the current section are available through the **Options** Menu in the Menu Bar. (See the section **Menu Bar** above.)

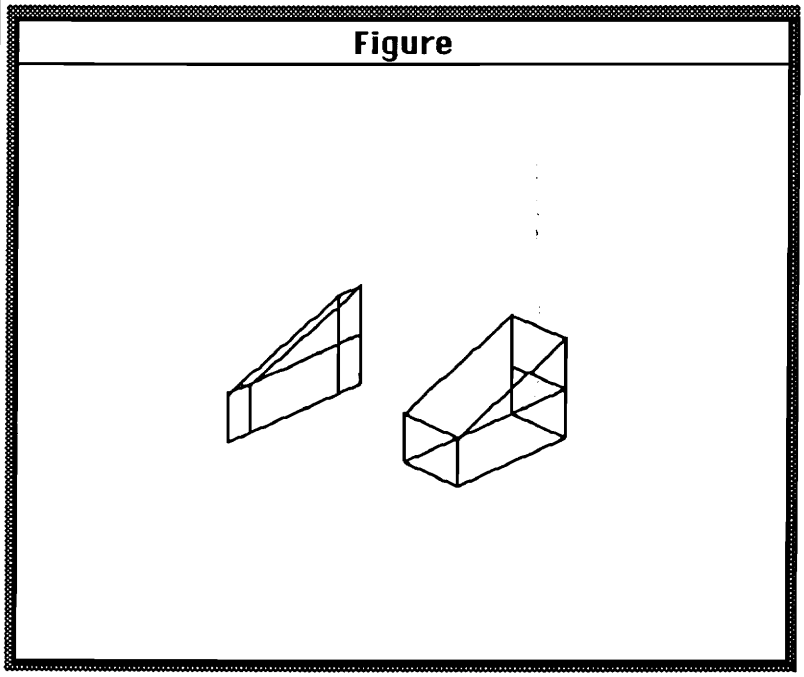


Figure 3. A screen dump showing an inactive Figure window.

Parameters

The **Parameters** window, which is actually a dialog box, contains parameter labels and their respective values, an **Apply** button, and a **Reset** button. Figure 4 shows a screen dump of an inactive **Parameters** window.

Labels are not editable, but the values are. You may edit a parameter value in the standard Macintosh manner.

When the user clicks with the mouse in the **Apply** button, the parameter values currently displayed are then applied to the figure.

When the user clicks in the **Reset** button, the parameter values are set to the default or original values, and the figure is modified accordingly.

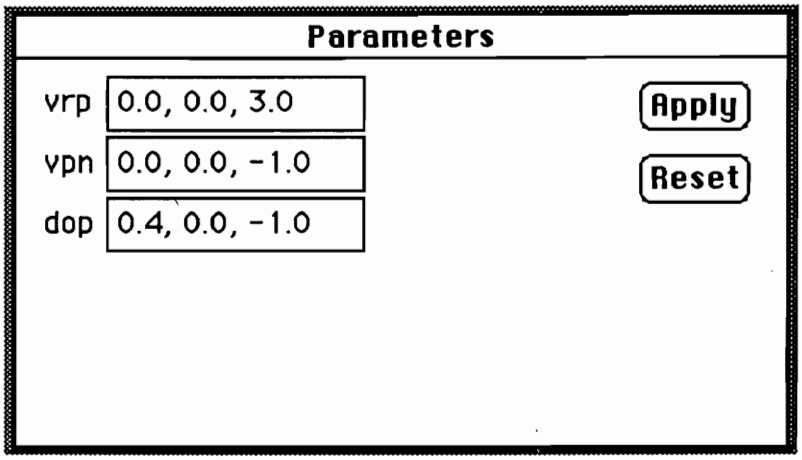


Figure 4. A screen dump showing an inactive Parameters window.

Glossary

The **Glossary** will be available only from a Section. To look for a word in the **Glossary**, first look for that word in the **Text** window. When you've found it, double-click on it. The **Glossary** entry of that word will appear in a dialog box. While you have the entry on the screen, you will not be able to do anything outside of the dialog box. When you are finished with the **Glossary** entry, click on the button marked **Done**. The dialog box with the **Glossary** entry will go away, and you may perform other activities in the Section.

DEVELOPER'S GUIDE

for the

INTERACTIVE COMPUTER GRAPHICS BOOK

by

Lucia I. Siochi

Computer Science Department
Virginia Tech
Blacksburg, VA 24061
siochil@csgrad.cs.vt.edu

April 1991

ABOUT THIS GUIDE

This is the Developer's Guide for the Interactive Computer Graphics Book (the ICGB, or the Book). The ICGB provides the user with an environment suitable for exploratory learning in the area of Computer Graphics. The Book is designed to allow expansion to cover other Computer Graphics topics not already present. This guide contains information that will help you add to or modify the Book.

Consult the README text file on the Developer's Disk for information not available at the time of this manual's printing. For more information on using the Book, see the User's Guide.

CONVENTIONS USED IN THIS GUIDE

This guide uses standard A/UX 2.0 and Macintosh OS terminologies.

Text in *italics* should be typed in as is. Text enclosed in <brackets> should be replaced by the appropriate text as described within the brackets. Key combinations may also be enclosed in brackets, for example, <ret> means "press the return key," and <control-l> means "press the control key and the letter 'l' at the same time."

Words appearing in **this font** refer to items in the application, such as menu items, window labels, and button labels.

TOC means "Table of Contents."

WHAT YOU WILL NEED

This chapter lists the items that you need to be able to expand the system.

System Configuration

You will need the following:

- a Macintosh II running
- A/UX 2.0 with
- the "mac32" environment set up.

Skills

The developer must have certain computer skills, called basic skills, to make even the simplest expansions. Other computer skills, called advanced skills, may be or are required for the more complex expansions.

Basic Skills

- familiarity with the UNIX or A/UX 2.0 programming environment
- familiarity with the mac32 environment

Advanced Skills

- understanding of the Computer Graphics topic being added
- ability to program in the C programming language
- familiarity with the Macintosh Toolbox ROM and its programming environment

Files

You will need the following text files in your working directory.

Make file

- makefile

Source files

- gt.c
- gt.r
- gt.windows.c
- gt.figures.c
- gt.menus.c
- matrix.c
- matrix.h
- macheaders.h
- gt.decls.h
- gt.r.h
- gt.utils.c
- gt.utils.h

Input files

- toc.txt

- *.txt
- *.fig
- *.prm

References

The following are references that you may need.

- This guide
- The README file in the Developer's Disk
- The C programming language by Kernighan and Ritchie, or another C reference
- Macintosh Revealed Volumes 1-4, Inside Macintosh Volumes I-V, or other Macintosh Toolbox references

INSTALLING THE DEVELOPMENT ENVIRONMENT

If any of the files listed below are missing, you must install the Book. This is done by making sure that these files are in your working directory on A/UX. Copies of these files are available on the Developer's Disk. Simply copy these files from the disk to your A/UX working directory.

These are the files that you will need to have in your A/UX working directory:

- makefile
- gt.c
- gt.r
- gt.windows.c
- gt.figures.c
- gt.menus.c
- matrix.c
- matrix.h
- macheaders.h
- gt.decls.h
- gt.r.h
- toc.txt
- *.txt
- *.fig
- *.prm
- gt.utils.c
- gt.utils.h

where * is a wild card.

ABOUT THE BOOK

As its name implies, this application is similar to a book. It has a Table of Contents, one or more Sections, and a Glossary. Each section has text and accompanying figures related to a particular topic.

However, the ICGB is more than just a static book. It provides you with an environment suitable for exploratory learning in the area of Computer Graphics. It automates the production and manipulation of figures that may be used to present and to clarify Computer Graphics subject matter.

This Book has the following parts:

- Table of Contents
- Section(s)
- Glossary
- Menu Bar

Each Section has the following parts:

- Text
- Figures
- Parameters

The implementation of the Glossary, however, is not yet completed.

ABOUT THE DEVELOPMENT ENVIRONMENT

The program runs under A/UX 2.0's mac32 hybrid environment. It uses C and the Macintosh Toolbox ROM, (the Toolbox, for short). It is designed to facilitate some modifications and additions. It is composed of the modules listed below, and uses various input files and include files.

- event module (gt.c, gt.decls.h) - This is the main program; it handles events and the event loop.
- menu module (gt.menus.c) - This takes care of most menu related data structures and routines.
- window module (gt.windows.c) - This takes care of most window related data structures and functions, and contains routines that manage the user's traversal within the "book." This module also takes care of defining and manipulating the contents of the text window.
- figure module (gt.figures.c) - This takes care of most figure and parameters related data structures and routines. Note that this module currently handles only figures for three-dimensional wireframe views and parallel projections.
- matrix module (matrix.c, matrix.h) - This contains matrix and vector operations for the figure module.

- utilities module (gt.utils.c, gt.utils.h) - This contains various routines useful for debugging.
- resource file (gt.r, gt.r.h) - This contains the resources and related information that the program uses.
- input files (toc.txt, *.txt, *.fig, *.prm) - These files include information about the Table of Contents' text, and about a section's text, figures, and parameters.

EXPANDING THE BOOK

To expand the Book, you will need to modify or to add some definitions and some links.

Definitions

This section discusses adding and modifying definitions.

While editing files, you may need to insert what are called "stickyspace" placeholders. For information on stickyspaces, refer to the section **The Stickyspace Character** in the chapter **TECHNICAL NOTES.**

Table of Contents

The text that appears in the **Table of Contents** window comes from the toc.txt text file. To add an entry <TOC-entry> to the TOC, edit the file toc.txt. Insert the text <TOC-entry> on a line by itself in between the appropriate TOC entries. Be sure to replace the spaces with a stickyspace placeholder. For examples, see the toc.txt file.

Section

Each section has definitions for its Text, Figures, and Parameters.

Text

The text that appears in section <sectionName>'s **Text** window comes from the toc.txt text file. To add a new section text definition for the section <sectionName>, create and edit the text file

<sectionName>.*txt*

Put in this file the text that you want to appear in <sectionName>'s Text window. Put the stickyspace placeholders in place of the spaces in the following:

- figure <figure-number>, and
- a phrase corresponding to a Glossary entry.

To modify a section's text, simply edit the respective .txt file, putting in stickyspace placeholders as described above.

Figures

The figure definitions of the section <sectionName> are in the text file <sectionName>.fig. Currently, only one figure type is supported. The format of the figure definitions file for this type is shown in figure 1.

Note that in the figure, <figurefile> refers to the contents of the text file. See the *.fig file(s).

Replace this:	with this:
<figurefile>	number of figures <F> <ret> figure 1 <figureinfo> <ret> figure 2 <figureinfo> ⋮ <ret> figure <F> <figureinfo>
<figureinfo>	paraminfo <prmdloginfo> <viewinfo> <metaviewinfo> <pointslist> <lineslist>
<prmdloginfo>	<resourceID#> 3 7
<resourceID#>	<an integer greater than 100 that is not already used by another figure>
<viewinfo>	vrp: <vrpx> <vrpy> <vrpz> vpn: <vpnx> <vpny> <vpnz> vup: <vupx> <vupy> <vupz> dop: <dopx> <dopy> <dopz> wmin: <wminx> <wminy> wmax: <wmaxx> <wmaxy> front: <f> back: <b>
<metaviewinfo>	<viewinfo>
<pointslist>	points <P> 0 <Pt0x> <Pt0y> <Pt0z> 1 <Pt1x> <Pt1y> <Pt1z> <...> <P-1> <Pt(P-1)x> <Pt(P-1)y> <Pt(P-1)z>
<lineslist>	lines <L> <Line1p1> <Line1p2> <Line2p1> <Line2p2> <...> <LineLp1> <LineLp2>

Figure 1. Format of the figure definitions in a .fig file.

Parameters

Parts of the parameters definitions are found in three text files, namely, <sectionName>.prm, gt.r.h, and gt.r.

Each figure needs a corresponding parameters definition. Part of the parameters definitions of the section <sectionName> are in the file <sectionName>.prm. The file template.prm contains the template for the definition of the parameters for one figure. This format is based on the format used for dialog box resources.

You will need as many working versions of this template as there are figures, one for each figure. All of this template's working versions for the figures of one Section are contained in this file, and nothing else is in the file. You may put any number of blank lines between each line in the file.

Each figure's parameters definition must have a number associated with it, which is located in gt.r.h. By convention, the numbers start at 128 for the first, and increase by 1 for each succeeding parameters definition. Each parameter in a definition must have a corresponding parameter label definition in gt.r.h.

The format is shown in figure 2a. You again need one working version of this template per figure. Currently, you are limited to three parameters for each parameters definition or dialog box.

```
/*

#define <sectionName>P<f> <128+f-1>
#define <prmNamep>ItmTxt
"<prmLabel1>"
#define <prmNamep>ItmTxt
"<prmLabel2>"
#define <prmNamep>ItmTxt
"<prmLabel3>"
*/
```

Figure 2a - Template for part of a parameters definition in the gt.r.h file.

For each Section, there should be a line in the file gt.r that has the format shown in figure 2b.

```
/*
#include <sectionName>.prm
*/
```

Figure 2b - Template for part of a parameters definition in the gt.r file.

Table 1 shows the changes to be made in the three files. Be sure to edit copies of the templates, and to remove

the outer comment characters from your working versions of the templates in figures 2a and 2b.

Table 1. Changes to make in <sectionName>.prm, gt.r.h, and gt.r for the Parameters definition.

Replace this	with this
<sectionName>	<the * of *.fig, *.txt, and *.prm files defining the corresponding section>
<f>	<unique integer of the corresponding figure in <sectionName>>
<prmNamep>	<the name of the macro defined in gt.r.h that corresponds to the label of the desired parameter, where p = 1, 2, or 3>
<prmLabel>	<the text that appears as the label for the corresponding parameter>

Glossary

Glossary entries will be defined in the text file glossary.txt. This file contains a sequence of glossary entries. Each entry will be preceded by a line with the string "entry" and has a name and text. The entry name must have the stickyspace placeholders replacing its spaces. The entry text follows the name immediately on the next line.

Links

Links should be set up between definitions as shown in figure 3. Definitions are represented by boxes or text. Links are represented by lines connecting definitions.

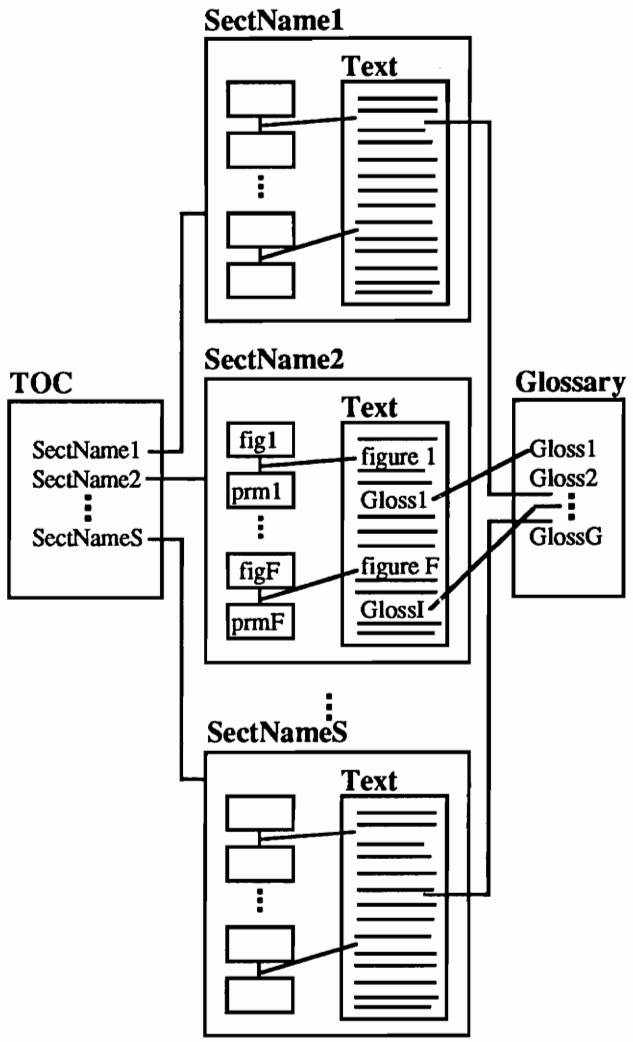


Figure 3. Definitions and the links between them.

The <TOC-Entry SectNameX> to <Section SectNameX> link

Edit the routine DoTocText() in gt.windows.c. Look for the comment as shown in figure 4. Insert a copy of this comment after the original comment, remove the comment characters, and edit the copy as shown in table 2.

```
/*
else if (strnequal (txt, "<TOC-
Entry>",<length>)) {
   if (OK2OpenASection()) {
      LeaveToc();
      OpenASection("<sectionName>");
      EnterSection();
      }
   }
*/
```

Figure 4. Template for the TOC to Section link.

Table 2. Changes to make in the routine DoTocText() in gt.windows.c for the TOC to Section link.

Replace this	with this
<TOC-Entry>	<the line of text in toc.txt that corresponds to the desired section>
<length>	<the number of characters in the <TOC-Entry> string, not counting '\0'>
<sectionName>	<the * of *.fig, *.txt, and *.prm files defining the corresponding section>

The <prmY> to <figY> link

Edit the routines ApplyChanges() and ResetParams(), both of which are in gt.figures.c. Look for the comments as shown in figures 5a and 5b. For each routine, insert a copy of the appropriate comment after the original comment, remove the comment characters, and edit the copy as shown in table 3. Add a pair of changes for each parameter that you want to link to the figure.

```
/*
else if (strequal (stItmTxt, <prmName>ItmTxt))
    String2Vector(&(<currfigprmStruct>),
etItmTxt);
*/
```

Figure 5a. Template in ApplyChanges() for part of one prm to fig link.

```
/*
else if (strequal (stItmTxt,
   <prmName>ItmTxt))
   Vector2String(etItmTxt,
   <initfigprmStruct>);
*/
```

Figure 5b. Template in ResetParams() for part of one prm to fig link.

Table 3. Changes to make in ApplyChanges() and ResetParams() in gt.figures.c for a prm to fig link.

Replace this	with this
<prmName>	<the name of the macro defined in gt.r.h that corresponds to label of the desired parameter>
<currfigprmStruct>	<the reference to the data structure containing the current value of the respective parameter>
<initfigprmStruct>	<the reference to the data structure containing the initial value of the respective parameter>

Rebuilding the Book

After editing the files, you may need to rebuild the Book. If you made changes to the input files only, you do not need to rebuild the Book, otherwise, you will need to rebuild. To rebuild the Book, type

make all <ret>

This may take a few minutes. (The gt.figure.c file takes longer to compile than the others.)

TECHNICAL NOTES

This chapter contains implementation-related information that you may find useful.

The Stickyspace Character

In the *.txt files, you may need to replace some blank spaces between words with a stickyspace character placeholder.

The stickyspace character has an octal ASCII value of 312 in the Macintosh character set, and looks like a blank space. This is useful for grouping words together, so that when the user double-clicks on a word in the group, the entire group is highlighted and selected.

Currently, <control-l>, (i.e., control-"ell"), is used as a placeholder in the *.txt files for the stickyspace character. When editing these files, put the placeholder where you want the stickyspace to be.

Case Sensitivity

Currently, the Book's implementation is case-sensitive. Be careful when setting up definitions and links; make sure that the corresponding text strings match exactly.

You may, however, consider stickyspaces as real blanks; they are converted to blanks prior to string comparisons.

Debugging Tools

When debugging, it can be useful to use some print statements. Printing to stdout (standard out) causes problems, but you may print to a file. The utilities module contains routines that print to gt.dbgout. You may use these routines or you may write your own.

Other Figure Types

The Book currently supports only figures with wireframe objects and 3D parallel projection views and metaviews. It was designed to facilitate the addition of sections using this figure type. However, many functions needed by all sections are already available, so it is possible to add sections that use other figure types. Since the Book was not designed to facilitate other figure types, doing so requires more than just adding lines of code from templates; it requires modifying the implementation.

To add other types of figures, you must modify the figures data structure to support your new figure type.

You will also need to add routines to carry out the computations necessary, possibly including the routine to draw the figure.

You will also need to modify one or more of the figure input routines such as GetFigDef().

If you define another figure type, or if you add a new parameter, you must keep track of two versions of the parameters: the current values, and the default or original values. The routines ApplyChanges() and ResetParams() need this information.