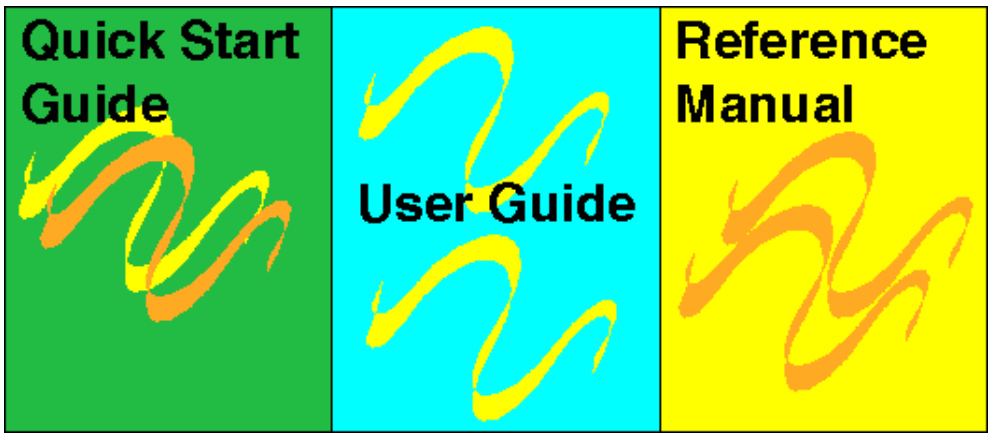
Welcome to the interactive documentation for NCAR Graphics Version 4

This is the user documentation for NCAR Graphics Version 4.0 (NG4).

You can access the three NG4 hypertext user documents from here:



The Quick Start Guide explains all NG4 examples and provides their source code.

The User Guide introduces NG4 software functionality and key concepts.

The Reference Manual describes how the NG4 software operates.

The User documentation for the traditional interface lists all the traditional documentation for the NCAR Graphics Low Level Utilities. This includes the programmer documents, user guide, and tutorials that give detailed descriptions for developing NCAR Graphics applications using the traditional interface.

The NG4 preface and the NG4 appendices provide other useful information about NCAR Graphics and its documentation.

Copyright 1995 University Corporation for Atmospheric Research
All rights reserved

$Revision: 1.4 $ $Date: 1995/07/20 14:59:03 $

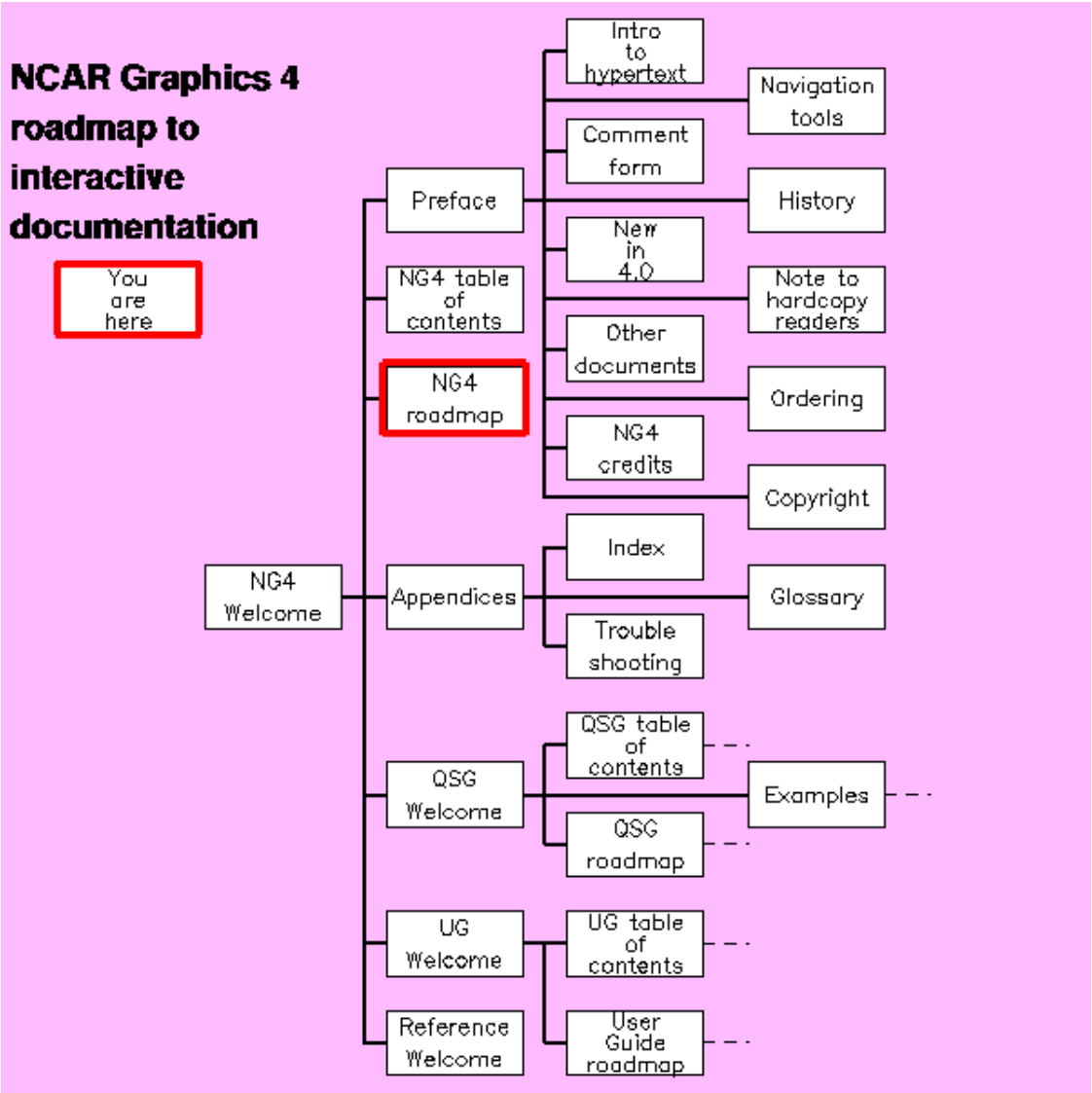
NG4 table of contents

Table of contents

NG4 welcome
- NG4 table of contents (this file)
- NG4 roadmap
- Preface
  - Introduction to hypertext documentation
  - Introducing the navigation tools
  - Comment form
  - History of NCAR Graphics
  - New features in Version 4.0
  - A note to our hardcopy readers
  - User documentation for NCAR Graphics Low Level Utilities
  - NG4 order form
  - NCAR Graphics credits
  - Copyright notice
- Appendices
  - Index
  - Glossary
  - Troubleshooting
- Quick Start Guide
- User Guide
- Reference Manual

$Revision: 1.9 $ $Date: 1995/07/26 15:56:33 $

NCAR Graphics 4 road map



$Revision: 1.4 $ $Date: 1995/07/13 19:05:20 $

Preface

NCAR Graphics software is documented in the interactive hypertext information system you are now using. The following modules provide the type of information you would find in the preface of a hardcopy book:

- Table of contents
- Roadmap
- Introduction to hypertext documentation

- Introducing the navigation tools
- History of NCAR Graphics
- New features in Version 4.0
- A note to our hardcopy readers
- Other user documentation for NCAR Graphics software
- NG4 order form
- NCAR Graphics credits

To continue on from this module, you can go to the:

- Quick Start Guide
- User Guide
- Reference Manual
- NG4 documentation appendix

$Revision: 1.3 $ $Date: 1995/07/19 19:07:15 $

Introduction to this hypertext documentation

This module describes some of the key features of the NG4 hypertext user documentation. This module does not explain much about how your browser presents information to you because there are many significant differences between the various browsers and even between different versions of the same browser. For detailed information about using your browser, use your browser's help feature.

Information modules are brief

Most of the modules in the NG4 user documentation are brief to avoid user fatigue from scrolling through large online files. Most modules deal with just one topic, which also improves their usability. Redundant information is avoided wherever possible: hypertext links allow relevant information to be accessed from anywhere in the system. This precludes the need to duplicate information. Finally, the links provided in the text and graphics have been planned to be as useful as possible to you; an excessive number of links and/or irrelevant links can interfere with comprehension of the material.

Orientation cues in color

The documentation for the three new interfaces to NCAR Graphics is divided into four parts. The NG4 preface files are visually coordinated for you with the color purple in the roadmap and in the control panels for each module. The Quick Start Guide is color-coded with green. The User Guide is color-coded with blue. The Reference Manual is color-coded with yellow. Through this color scheme, every module in the NG4 hypertext documentation provides cues to keep you oriented.

Navigation tools in every module

Every module offers a control panel at its beginning and end. These control panels allow you to move efficiently from one area of the information system to another. While this is advantageous in many ways, you might also become confused about where you are and where you came from. To minimize such confusion, we have provided color cues and roadmaps. (These navigation tools are discussed in the preceding paragraph.)

Navigation tools in your browser

You can also use the navigation tools available from your browser to help you move around the documentation. A "Back" button returns you to the module you were viewing prior to your currently displayed module. You can use this button repeatedly to retrace your path through the hypertext files. There is also a "Forward" button that becomes active after you use the Back button.

The Netscape browser has a "Go" menu that shows you a list of all the files you have visited; you can select a file directly from this list without having to repeatedly use the Back or Forward buttons. The Mosaic browser provides this same functionality in the "Navigate" menu, under "Window History."

An extremely useful tool provided by your browser is the "Bookmark" (Netscape) or the "Hotlist" (Mosaic). This feature allows you to record the location of any file with the click of a button: you can use this to keep track of modules that you will need to visit often. Netscape allows you to organize these locations under headings, and Mosaic records your selections chronologically (new locations are added to the bottom of the list).

> Of course, you can create your own HTML document--a personalized "home page"--in which you can organize your favorite locations in ways that are most useful to you. Many people "publish" their home pages on the World Wide Web (WWW) to publicly express their creativity and individuality. For more information about the World Wide Web, you might find value in the WWW Frequently Asked Questions document.

Ongoing documentation updates

The NG4 documentation on the NCAR server is being continually upated. The files you are viewing on your local server were "frozen" at the time we wrote them onto the distribution medium (compact disk or magnetic tape). As of the 4.0 release, NCAR Graphics staff members were discussing ways to provide our customers with automatic updates of their local NG4 documentation files, but such a system has not been implemented as of the summer of 1995.

So if you want to check for a more up-to-date version of any file, add http://ngwww.ucar.edu/ngdoc/ng4.0/ngdochome.html to the "Bookmarks" or "Hotlist" in your browser. Then visit our server for the most current documentation available.

Printing hardcopy

If you find that you use certain modules frequently, you may decide that it is worthwhile to print those modules. You can print any module using the "Print" function in your browser’s "File" menu. This approach can be more convenient that having a complete hardcopy manual at your desk, but your printed modules may become outdated as the NG4 user documentation evolves. You can also print a

copy of the latest documentation from NCAR's server specified in the preceding paragraph.

PostScript copies of fully-assembled NG4 user documents have been provided on the NCAR Graphics distribution. These are not designed as hardcopy manuals; they are concatenated, in table-of-contents order, from the hypertext files.

$Revision: 1.8 $ $Date: 1995/07/26 05:01:45 $

Introducing the navigation tools

The NG4 navigation tools appear at the beginning and end of every information module.

Control panels

In the graphical control panel at the beginning of each module, and in the roadmaps that you can access from anywhere in the NG4 documentation, the background color serves as an orientation tool: it indicates your general location in the documentation:

- Purple for the home page, preface, and appendix
- Green for the Quick Start Guide
- Blue for the User Guide
- Yellow for the Reference Manual

NG4 home Click on this button to return to the NG4 home page.

Index Click on this button to open a new window that shows an index similar to the index you would expect at the end of a technical manual. Rather than listing page numbers, however, each entry in the index is a link to one location in the information system. Unlike the roadmap and contents tools, the index covers all three documents (User Guide, Quick Start Guide, and Reference Manual), not just the current document.

Examples Click on this button to open a new window that shows the utilities that have HLU, NCL, and/or GUI examples available. Click on one of these utilities to see images of all examples available for that utility. Click on one of the examples to see an explanation of how the code produces

that example.

Glossary Click on this button to view the glossary of terms and acronyms used in NG4. Use your browser's "back" button to return to your previous location.

Table of Contents Click on this button to open a new window that shows the table of contents for the current document (either the User Guide, the Quick Start Guide, or the Reference Manual). Each table of contents shows all headings in the document, organized as they would be if this were a conventional printed book. Click on any heading in the contents to move to that module.

Feedback Click on this button to open a form that allows you to submit comments, questions, and requests about the documentation and the NG4 software.

Roadmap Click on this button to see a graphical representation of how the information modules in the current document are organized (either the User Guide, the Quick Start Guide, or the Reference Manual).

You can click on any module shown on the roadmap graphic to move to that module. The roadmaps have been subdivided to make them small enough for use on normal-sized workstation screens.

The highlighted box on the roadmap graphic indicates which module you were just viewing. The background color indicates your general location in the interactive documentation:

- Purple for the home page, preface, and appendix
- Green for the Quick Start Guide
- Blue for the User Guide
- Yellow for the Reference Manual

Hypertext links

Throughout the text and graphics that make up the NG4 documentation modules, hypertext links are provided to lead you to additional useful information. These links are highlighted in a way that is determined by your browser (Netscape, Mosaic, Lynx, etc.) and the computer system on which it is running.

To avoid providing too many links (which quickly becomes distracting), we have made every effort to place links only where they will be relevant to what we believe you might need as you use the documentation.

$Revision: 1.8 $ $Date: 1995/07/26 04:03:22 $

Send us your comments on NCAR Graphics documentation

You can also use this form to make enhancement requests.

Thank you for taking the time to send us your comments and suggestions.

We are interested in any feedback you care to send, but we are specifically concerned about correcting any factual errors and unclear passages in the documentation. Please type in your comments and/or requests here or send email directly to ng4doc@ncar.ucar.edu. Please include the module name if your comments are specific to a particular module:

Click here to edit and send us your comments.

$Revision: 1.8 $ $Date: 1995/07/14 01:28:28 $

History of NCAR Graphics

The NCAR Graphics package has a long history and a bright future.

Decades of development

The NCAR Graphics package has been evolving since its inception in the early 1960s as a collection of low-level Fortran routines for drawing simple graphical elements on the output devices of that era.

In the 1970s, NCAR scientists and Scientific Computing Division staff developed higher-level graphics utilities which were integrated into a package that could be distributed throughout the scientific

community. Portability was enhanced as the utilities generated an output metafile to acheive device independence.

In the 1980s, still more functionality was added, and the whole package was brought into conformance with the graphics and programming language standards that emerged at that time. In late 1986, the updated set of the NCAR Graphics utilities was converted to Fortran 77, the metafile was based upon a private encoding of the Computer Graphics Metafile (CGM) standard instead of the early home-grown metafile format, and the utilities were moved to the Graphical Kernel System (GKS) standard. This standards-conforming package has been distributed to about 1000 sites worldwide.

In the 1990s, the functionality of NCAR Graphics has been growing, both from the expanded power provided by GKS and from the addition of new utilities. With the release of NCAR Graphics Version 3.2 in 1993, a second programmatic interface was added: C program bindings were included for all calls to all utilities. NCAR Graphics is widely viewed as a mature two-dimensional visualization package.

New interfaces for the future

The traditional Fortran interface to NCAR Graphics can be difficult to use because it does not have a consistent user interface. This is because the various utilities were developed by numerous people through many years. Its components have been collected into one package, but new users of the package are often confused by the differences in the Fortran calls needed to control the various utilities.

The package is robust, it is widely used in the scientific community, and it provides a wide range of 2-D visualization capabilities. Rather than pursuing more 2-D or 3-D functionality, the NCAR Graphics Group is now focusing its efforts on improving the user interface.

This is why *three* new interfaces are being developed: High Level Utilities (HLUs), NCAR Command Language (NCL), and Graphical User Interface (GUI). When all three are fully implemented, NCAR Graphics will offer five user interfaces to all of its functionality.

The functionality in NCAR Graphics Version 4.0 is almost identical to that of Version 3.2, that is, the Fortran and C programmatic interfaces have not changed much.

The significant difference is that two new interfaces are partially implemented for a subset of the NCAR Graphics functionality. The High Level Utilities (HLUs) and the NCAR Command Language (NCL) are the beginnings of a consistent visualization model for all of NCAR Graphics. The Graphical User Interface (GUI) is being released as an unsupported prototype at Version 4.0.

In future releases, all NCAR Graphics functionality will be accessible through all three new interfaces.

NCAR Graphics developers have always strived to maintain compatibility between the many versions and releases of the NCAR Graphics package. It is our intention to keep supporting--for as long as possible--older user programs that were written for earlier versions of the package.

At Version 4.0, you should find that your existing Fortran and C programs that call routines in the NCAR Graphics libraries will continue to function as they always have.

If you notice functional discrepancies in the way your existing codes perform, you should contact your

local site representative, and your site rep will determine if NCAR Graphics or SCD Consulting staff needs to be notified.

You might also consider joining an e-mail group of NCAR Graphics users and developers where a wide variety of usage issues are discussed and resolved. This group is called "ncarg-talk", and you can join by sending an e-mail message to Majordomo@UCAR.EDU with `subscribe ncarg-talk` in the *body* of the message (not the subject line). You can also contact a human by sending e-mail to ncarg-talk-approval@UCAR.EDU.

$Revision: 1.8 $ $Date: 1995/07/26 07:25:03 $

New features in Version 4.0

This module provides a list of new features that are being released in NCAR Graphics Version 4.0. It also specifies the platforms on which these features will be available at 4.0.

LLU interface

- Available on all supported platforms.
- This is the familiar NCARG 3.2 with new features and improvements.
- New features:
  - Areas: Added new algorithms for reconciling conflicting area-identifier information which work better for Conpack-generated area maps (see the parameter 'RC'). Debug plots can now be made to show all edge groups.
  - Dashpack: Draws curves using dashed-line patterns that may include gap-portion specifiers, solid-portion specifiers, and label-string specifiers. Labels can be made to bend with the curve. Dashpack is intended to replace and unify the old family of NCAR Graphics routines that includes Dashline, Dashchar, Dashsmth, and Dashsupr.
  - Ezmap: The new routine MAPBLM draws continental/political outlines masked by the contents of an area map. (In 4.0.1, use of a logical unit by Ezmap will be avoided.)
  - Labelbar: Modified so that all-blank labels are just ignored when it picks the character size to be used; what this means is that, if every other label is blank, the remaining ones will be made bigger.
  - Plotchar: Added a routine PCRSET, which returns all internal parameters to their default values. The new internal parameter 'UN' determines what logical unit will be used to access the PWRITX dataset. (In 4.0.1, use of a logical unit will be avoided entirely.) Added a mapping capability that is useful for generating labels on maps. Added parameters 'ZX', 'ZY', and 'ZZ' to specify default "zoom" values.
  - Polypack: Allows users to manipulate polygons in various ways.
  - Wmap: Aids in the production of daily weather maps like the ones you see in newspapers: plots station model data including wind barbs and draws icons to indicate daily weather.
  - Direct PostScript output: Now generates PS files directly (in addition to NCGMs). Types of PS include PS, EPS, and limited EPSI in color, mono, portrait, and landscape.
  - General: There is a new example for AREAS/CONPACK/EZMAP/PLOTCHAR, illustrating

new features.

- Improved features:
  - Areas: Fixed some bugs, one of them probably responsible for some of those annoying "ALGORITHM FAILURE" exits.
  - Conpack: Fixed some bugs. Added an interface to Dashpack. Installed a new option to use interpolation only in problem areas (see the new parameter 'PIT'). Improved the generation of the visible/invisible boundary in an area map, thus avoiding certain problems with color fill.
  - Ezmap: Fixed an error that kept MAPGRM from responding properly to the color-setting parameters. Fixed an error affecting the handling of the "pole longitude".
  - Plotchar: Fixed a bug that prevented the use of filled fonts on Cray machines. Made it possible to use two consecutive occurrences of the function control character to get that character output as itself.
  - General: Improved error handling (recovery, traceback, etc.).

New interfaces

- There are three new user interfaces: HLU, NCL, and a prototype GUI.
- HLU = High Level Utilities.
  - Available on Sun computers, SGIs, and others. (Cray Unicos in 4.?)
  - New programming language interface (for Fortran and C).
  - Uses an object-oriented paradigm.
  - Allows use of "resource files" to control options.
  - Supports these classes of objects: XyPlot, Contour, Map, TextItem, LabelBar, Legend, Title, TickMark, Annotation
  - Future classes (projected for 4.1): Histogram, StreamLine, Vector, 3D
- NCL = NCAR Command Language.
  - Available on Sun computers, SGIs, and others. (Cray Unicos in 4.?)
  - An interpreted command language.
  - Uses some language constructs of Fortran 90.
  - Supports multiple input data formats.
- GUI = Graphical User Interface.
  - Available on all of the regular supported systems (Digital Equipment Alpha (OSF/1 3.2), Digital Equipment DECstation (Ultrix 4.4), HP9000 Series 700 (HP-UX 9.01), IBM RISC System/6000 (AIX 3.2), Silicon Graphics R4 systems (IRIX 5.3), and Sun-4/SPARCstation (SunOS 4.1)) except the Cray and the Sun4 Solaris.
  - GUI is an unsupported prototype at 4.0.

Online documentation

- LLU interface
  - Hypertext: http://ngwww.ucar.edu/ngdoc/ng4.0/ug/llu/lluoview.html
  - man pages
  - Examples via ncargex/ncargcex
  - *Fundamentals*, *Contouring and Mapping Tutorial*, *GKS User Guide* and other Low Level Utility programmer documents (many are vailable in HTML format and PostScript).
- New interfaces

- Hypertext: http://ngwww.ucar.edu/ngdoc/ng4.0/ngdochome.html
- Examples via ng4ex
- User Guide, Quick Start Guide, Reference Manual

$Revision: 1.5 $ $Date: 1995/07/24 16:56:18 $

A note to our hardcopy readers

The NCAR Graphics Version 4.0 documentation was designed as an online hypertext information system to be read using a hypertext browser like Netscape or Mosaic. We recommend viewing the documentation using a hypertext browser rather than reading the documentation in a hardcopy form. Depending on the browser you use, you can conveniently navigate through the documentation by just clicking on highlighted text and/or using the control panel tools that appear at the top and bottom of every module.

Forced to use hardcopy?

However, if you are unable to use a hypertext browser, you will be constrained to using the PostScript versions of the User Guide, the Quick Start Guide, and the Reference Manual. You will lose many of the hypertext advantages offered by the interactive documentation, and you may have to wait for new software releases to obtain documentation updates.

We've placed PostScript copies of the hypertext user documentation in the ngdoc/ng4.0/nggenrl/psdocs directory. Note that some of the .ps files have been compressed with the UNIX compress command, making them .ps.Z files. However, the installer at your site may have elected not install these large files on your local system because they require so much storage space. If the links in the following paragraph do not work for you, and if you must have the hardcopy versions of this interactive documentation, then you need to seek the assistance of your system administrator.

The PostScript versions of the NG4 preface files, User Guide, Quick Start Guide, and Reference Manual may not be installed on your system. The PostScript files of the manuals for the traditional NCAR Graphics Fortran and C interfaces, *NCAR Graphics Fundamentals*, *NCAR Graphics Contouring and Mapping Tutorial*, and *User's Guide for NCAR GKS-0A Graphics*, may likewise be missing from your local computer.

The programmer documents covering each of the utilities for the traditional programmatic interfaces are available in PostScript form. You can expect to find them in the locations described by the module User documentation for NCAR Graphics Low Level Utilities.

How to obtain a hypertext browser

If you are using a graphics workstation, we recommend using a full-screen hypertext browser like Netscape or Mosaic. These browsers are currently free, however, we cannot distribute them with the

NCAR Graphics software. If your workstation does not support graphics, then we recommend using a line mode browser like Lynx.

Feedback helps everyone

If you have any difficulty with the documentation, please send your comments and suggestions using the Feedback button on the control panels at the beginning and end of every module, or contact us at one of the following addresses:

ng4doc@ncar.ucar.edu - Email address

(303) 497-1201 - NCAR Graphics Information Line

SCD/NCAR Graphics Documentation
University Corporation for Atmospheric Research
P.O. Box 3000
Boulder, CO 80307-3000 (USA)

$Revision: 1.10 $ $Date: 1995/07/26 14:23:07 $

User documentation for NCAR Graphics Low Level Utilities

To use traditional NCAR Graphics (the Low Level Utilities), you write a Fortran or C program to call NCAR Graphics routines that produce visual representations of your data. Traditional NCAR Graphics is thoroughly documented in:

- The introductory user guide *NCAR Graphics Fundamentals*
- The training manual *NCAR Graphics Contouring and Mapping Tutorial*
- The NCAR Graphics GKS manual *User's Guide for NCAR GKS-0A Graphics*
- The full set of NCAR Graphics man pages
- Examples that show NCAR Graphics capabilities and help you to understand and use the code that produces the examples
- Programmer documents for each Low Level Utility, written by the software developers. Note: These programmer documents are currently only available in PostScript format.
  - Areas, A Package that Solves Certain Practical Problems in Topology
  - Autograph, A Package of Routines to Draw X/Y Plots
  - CGM, Graphcap, and Fontcap Supplement
  - Conpack, A Contouring Package
  - Dashpack, A Software Package for Drawing Dashed Lines
  - Ezmap, A Map-Drawing Package
  - Gflash, A Graphics Instruction Manipulation Package
  - Gridal, A Package to Draw Backgrounds for X/Y Plots

- Isosurface, A Package of Routines used to Draw Isosurfaces
- Labelbar, A Package of Routines to Draw Labelled Bars
- Plotchar, A Character-Plotting Package
- Polypack, A Package of Routines to Manipulate Polygons
- Stitle, A Package to Create Scrolled Movie Titles
- Seter, A Centralized Error-Handling Package for NCAR Graphics
- Softfill, A Software Fill Package
- Streamlines, A Field Flow Visualization Utility
- Vectors, A Vector Field Plotting Utility
- Wmap, A Package for Producing Daily Weather Maps and Plotting Station Model Data

NCAR Graphics Fundamentals contains important information for all users of traditional NCAR Graphics: a glossary of terms, explanation of the structure of NCAR Graphics programs, strategies for learning to use NCAR Graphics, instructions for compiling, linking, and running programs, instructions for producing and editing output from NCAR Graphics programs, and guidelines for using the utilities. It introduces the Fortran interface to all of the traditional NCAR Graphics utilities except for Areas, Ezmap, and Conpack, which are explained in full detail in the hardcopy manual *NCAR Graphics Contouring and Mapping Tutorial*.

The *NCAR Graphics Contouring and Mapping Tutorial* is a learning tool for beginning and intermediate users of the Fortran interface to NCAR Graphics. It gives examples and describes how to use all routines in the Areas, Ezmap, and Conpack utilities. Users combine these routines to prepare NCAR Graphics programs that fill areas, draw maps, and contour data fields.

The online man pages provide concise reference information for all user entry points, user-modifiable internal routines, parameters used in all utilities, and NCAR Graphics commands such as **ncargex**. The overview man page **ncargintro** introduces all utilities in the package, and overview man pages for each utility introduce all routines in that utility. The man pages should be installed on all systems that runNCAR Graphics software.

Programmer documents are the software developers' reference notes; they are provided for users who want to explore the information that developers use to maintain a utility's source code. They can also be very helpful to users.

The **ncargex** command allows you to access more than 250 examples, and all of them are internally documented with comments. A brief introduction to **ncargex** is provided in the *Tutorial* module Intro 1.2 "Details about how information is presented in this manual,"and procedures for using **ncargex** appear in the *Fundamentals* " Chapter 2: Five quick steps for creating and viewing your plot." A pictorial index to all NCAR graphics examples is provided as Appendix D of *Fundamentals*; this pictorial index appears only in the hardcopy version of the manual, and the hardcopy manual is likely to become unavailable sometime during 1995. While supplies of the printed manuals last, one copy each of *Fundamentals* and *Tutorial* is shipped to sites that order any version of NCAR Graphics at Version 3.2 or later.

Additional special-purpose user documents are available via anonymous FTP to ftp.ucar.edu; see the **ncarg** directory. These documents include guides to producing films and videos using NCAR facilities and a reference guide to the NCAR CGM format. The complete list of these special-purpose documents appears in the online NCAR Graphics order form

$Revision: 1.6 $ $Date: 1995/07/26 02:16:20 $

Ordering NCAR Graphics software

You can request information about the NCAR Graphics software package by
phone:

- (303) 497-1201

e-mail:

- scdinfo@ncar.ucar.edu

or **Postal Service mail:**

- NCAR Graphics/Orders
  P.O. Box 3000
  Boulder, CO 80307-3000

Or you can request your packet of information by using this form:

NCAR Graphics software information
Order form

Please enter as much information about yourself as necessary,
specify how you want the information sent to you,
then click on the "Submit request" button at the bottom.

- `Name:`
- `Department:`
- `Organization:`
- `Street address:`
- `Address 2:`
- `City, State, Zip:`
- `Country:`
- `Phone number:`
- `FAX number:`
- `E-mail:`

Please send my information about ordering NCAR Graphics software via:

E-mail (to the address above)
Postal Service mail
FAX

Any additional comments?

$Revision: 1.4 $ $Date: 1995/07/26 06:02:22 $

NG4 documentation credits

Acknowledgments

Authors: Ethan Alpert, Brian Bevirt, Jeff Boote, Dave Brown, Fred Clare, Quindi Franco, Mary Haley, Dave Kennison, Bob Lackman, Jacque Marshall, Tom Parker, Tim Scheitlin, Ed Stautler

Editor: Brian Bevirt

Production: Jacque Marshall

Copyright

Copyright © 1995 University Corporation for Atmospheric Research
All Rights Reserved

Trademarks

All brand and product names are trademarks or registered trademarks of their respective holders. Reference to a company or product name does not imply approval or recommendation of that company or product to the exclusion of others.

Published by

National Center for Atmospheric Research, Scientific Computing Division, P.O. Box 3000, Boulder, CO 80307-3000. The National Center for Atmospheric Research is operated by the University Corporation for Atmospheric Research and is sponsored by the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

$Revision: 1.4 $ $Date: 1995/07/19 17:14:47 $

Copyright notice

Copyright 1995 University Corporation for Atmospheric Research (UCAR)
National Center for Atmospheric Research (NCAR).

This documentation was developed by NCAR, which is operated by UCAR and sponsored by the National Science Foundation.

Access and use of this documentation shall impose the following obligations on the user. The user is granted the right, without any fee or cost, to use or reproduce this documentation provided that this entire notice appears in all copies of the documentation. Further, the user agrees to credit UCAR/NCAR in any publications that result from the use of this documentation. The names UCAR/NCAR, however, may not be used in any advertising or publicity to endorse or promote any products or commercial entity unless specific written permission is obtained from UCAR/NCAR. The user also understands that UCAR/NCAR is not obligated to provide the user with any support, consulting, training or assistance of any kind with regard to the use of this documentation nor to provide the user with any updates, revisions or new versions.

THIS DOCUMENTATION IS PROVIDED BY UCAR/NCAR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UCAR/NCAR BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO CLAIMS ASSOCIATED WITH THE LOSS OF DATA OR PROFITS, WHICH MAY RESULT FROM AN ACTION IN CONTRACT, NEGLIGENCE OR OTHER TORTIOUS CLAIM THAT ARISES OUT OF OR IN CONNECTION WITH THE ACCESS, USE OR PERFORMANCE OF THIS DOCUMENTATION.

$Revision: 1.3 $ $Date: 1995/07/24 17:30:07 $

NG4 documentation appendix

The following modules provide the type of information you would find in the appendix of a hardcopy book:

- Troubleshooting NG4 usage problems:
  - General

    - HLU
    - NCL
    - GUI
    - LLU
- Glossary
- Index
- 4.0 Release Notes
- 4.0 FAQ

$Revision: 1.8 $ $Date: 1995/07/26 16:16:12 $

Glossary of NCAR Graphics terms

Definitions that are specific to a particular part of the package are prefaced accordingly. For example, definitions that apply only to NCL are prefaced with **NCL:** .

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

A

---

annotation
: A viewable object whose location and usually size are set relative to the viewport or data coordinate space of a base plot. There are three kinds of annotations: intrinsic annotations, embedded annotations, and external annotations.

annotation functions
: The functions used to add and remove annotations to and from plot objects. Specifically, these functions are NhlAddAnnotation and NhlRemoveAnnotation for the C and Fortran interfaces.

annotation plot
: An annotation that is a plot object and not simply a viewable object. An annotation plot is a subordinate base plot.

ANSI
: The American National Standards Institute, an independent non-profit organization that creates and publishes U.S. national standards (such as Fortran, C, CGM, and so forth) taking input from all sectors of the technical community and the public at large. ANSI also works in collaboration with other standards organizations such as ISO (the International Standards Organization) and IEEE (the Institute of Electrical and Electronics Engineers).

API
: Application Programming Interface.

application
: A program written in C, Fortran, or NCL that utilizes any of the functionality of NCAR Graphics.

application class

Refers specifically the class App. Objects that are instances of this class are used to keep track of resource databases. Every application must create at least one App object (this will be done automatically for you if you use NhlOpen).

Application Programming Interface

The programming interfaces (C, Fortran, and NCL) to the NCAR Graphics package. These interfaces provide access to the support functions defined in the classes as well as provide additional useful functionality.

application resource file

A resource file that is specific to a particular application. There are two application-specific resource files: a system application-specific resource file and a user application-specific resource file. The user can specify what directories the application-specific resource files are in. By default, the user application-specific resource file is in the local current directory and the system application-specific resource file is in the directory specified by the setting of the environment variable NCARG_SYSAPPRES. Resources defined in the user application-specific resource file will override resources defined in the system application resource file.

area fill pattern

A pattern to use for filling a polygonal area. The patterns are selected by using an integer fill index into a table of patterns.

ASCII

Stands for "American Standard Code for Information Interchange." This is an ANSI Standard specifying a set of 128 characters with their associated coded integer representations.

ASCII file

NCL: A data file that contains integers or floating point data values in ASCII format.

aspect ratio

Specifies the height-to-width ratio of a plot. This term is also applied to characters. For example, characters with an aspect ratio of 2.0 are twice as tall as they are wide.

associative operator

A binary operator that obeys the law of associativity: i.e. a binary operator "R" such that (aRb)Rc = aR(bRc) for all legal operands a, b, and c.

attribute

NCL: A singly-dimensioned datum of any type that is assigned to a variable using the ' @ ' operator. An attribute of a variable contains descriptive information about the variable.

B

---

background color

The color that will be used as a background color for the entire viewable surface of a physical workstation when plots are drawn on it.

base plot

A plot object responsible for setting the viewport of zero or more plot members relative to its own viewport. There are two kinds of base plot: primary base plots and subordinate base plots. At creation, any plot object is a primary base plot. A plot object ceases to be a base plot when added to another plot object as an overlay. When added as an annotation a plot object becomes a subordinate base plot. A plot object must be a primary base plot for users to draw it or change its workstation.

binary file
NCL: A data file that contains integer or floating point data stored in a format not supported by NCL.

bounding box
For View class objects, the bounding box for such objects is the smallest rectangle in NDC space that contains all of the marks that would appear on an output workstation if the object were drawn.

C

CGM
Computer Graphics Metafile.

child
A relationship that holds between *objects*. If "A" and "B" are objects, then B is a child of A provided that when B was created, A was specified as being its parent (either in the fourth argument of an NhlCreate call, or in a NCL create expression), or B was made a child of A by using the NhlChangeWorkstation function. If B is a child of A, then the following conditions apply:

1. B inherits the resource database of A. If viewable, B will display to the same workstation as A; if A is a workstation, then B will draw to A.
2. Destroying A will destroy B.
3. Resources can be specified in resource files as: {App obj name} . {parent of A} . {Name of A} . {Name of B} . {resource of B} : {value}.

A child can have only one parent. See also parent.

child/parent hierarchy
The tree structure determined by the child/parent relationship existing among all current objects in an application. A child can have only a single parent, but a parent may have many children. The child/parent hierarchy should not be confused with the class hierarchy. See also class hierarchy.

class
A template for defining objects that specifies variables, and procedures that operate on those variables. In the context of the NCAR HLU library, the class variables are called resources and the class procedures are called support functions. Objects are members, or instances, of a class formed by assigning specific values to the variables in the class.

class hierarchy
Each class, except the base class, is derived from some other class. The tree structure determined by the derived-class/superclass relationships among all of the classes is called the class hierarchy. The class hierarchy should not be confused with the child/parent hierarchy.

color index
See color table.

color map
Same as color table.

color table
A table that associates integer values (called color indices) with RGB color values. In NCAR Graphics, color tables contain up to a maximum of 256 colors (including the background color).

command
NCL: Same as an NCL statement.

composite class
A class that combines the resources of other classes with its own. A composite class inherits resources and functions from its superclass and it shares the resources from its composite members by the process of resource forwarding. If the composite class members have support functions, these functions do not apply to the composite class.

composite class member
A class used as part of the functionality of a composite class.

composite class resource
A resource available to a composite class by way of resource forwarding from a member class of the composite class.

Computer Graphics Metafile
A graphics metafile is a file that contains encoded vector graphics elements such as lines, colors, dash patterns, markers, and so forth. The Computer Graphics Metafile (CGM) is a precisely-defined formatting for a graphics metafile as defined and standardized by ANSI. NCAR Graphics produces a version of the CGM that is called a *conforming private encoding* that can easily be converted to and from standard CGM by using the filters ncgm2cgm and cgm2ncgm. The NCAR private encoding is also called NCAR CGM, or NCGM.

coordinate variable
NCL: A value associated with a named dimension of a variable or file variable that contains numerical coordinate information for each index of the dimension. Coordinate variables must be singly- dimensioned values. Warnings are produced if the array of values assigned is not monotonically increasing or decreasing.

ctrans
The NCGM interpreter distributed with NCAR graphics.

D

---

dash pattern
A pattern (such as "solid", "dotted", and so forth) to use as a line style when lines are plotted using the draw function. Dash patterns are selected by using an integer index into a table of dash patterns.

data classes
Any of the classes that are used to provide user input data to any of the objects that utilize such data. These classes are the CoordArrays class, the CoordArrTable class, and the ScalarField class.

data conversion
The process of converting data stored in one format to another format, such as converting data stored as integers to data stored as floating point numbers. Some objects, such as ScalarField, perform automatic data conversions.

data coordinate space
The coordinate space that is appropriate to input data. Transformations can be effected between data coordinate space and NDC (see Normalized Device Coordinates).

data specific resource
A resource of a particular class, such as the XyPlot class, that can be used to modify the attributes

of data supplied via a DataSpec object. Data specific resources can be used to control attributes such as curve colors, dash patterns, marker sizes, marker colors, and so forth.

data transformation
A process that transforms data from one coordinate space to another, such as transforming data in logarithmic space to data in linear space.

data type
NCL: A data type is a representation of data that defines a size and valid range for numerical data or provides a reference to a file or HLU graphical object.

derived class
See subclass.

drawable object
See viewable object.

draw function
Specifically, either the NhlDraw function of the C or Fortran interfaces, or the draw function of NCL, that is invoked to plot a View object.

E

---

embedded annotation
An annotation that may be incorporated as part of the functionality of subclasses of the Transform class. It is managed internally by the controlling PlotManager.

Encapsulated PostScript
Encapsulated PostScript (EPS) is a subset of regular PostScript. The restrictions placed on EPS files are for making it an appropriate format for importing into applications that import PostScript.

Encapsulated PostScript Interchange Format
Encapsulated PostScript Interchange Format (EPSI) files are Encapsulated PostScript files that have a "preview bitmap" that represents the PostScript image contained in the file. The bitmap (and it is a bitmap and not a color map) can be used by an importing application to quickly display a picture of the imported file.

EPS
see Encapsulated PostScript

EPSI
see Encapsulated PostScript Interchange Format

error class
A class that is used to configure the error reporting module of the HLU library. For any application, there is precisely one error object created, and it is created automatically for you. The error class defines several resources for controlling error reporting.

expression
NCL: Any sequence of NCL operators and operands that results in the computation of a value. In particular, any literal value is an expression and any variable is an expression. Also, arrays are expressions as well as functions. Operators applied to expressions are expressions.

external annotation
An annotation consisting of an arbitrary user-created viewable object added to a plot object. The user controls the location and size relative to the base plot by manipulating the resources of a user-accessible AnnoManager object.

F

file
: **NCL:** A data file residing external to NCL in one of NCL's supported data formats.

file variable
: **NCL:** A variable, created by the NCL addfile function, that contains a reference to a file.

fill value
: **NCL:** Same as missing value.

fontcap
: A file that contains detailed information used to plot characters. Fontcaps have a human-readable ASCII form and a binary form that is readable by ctrans.

foreground color
: The color associated with color index 1. This is used as the default color in drawing viewable objects.

function
: **HLU:** Any member of the HLU API.
NCL: An identifier with a list of parameters separated by commas and enclosed in parentheses. Functions return values when called. A function is defined by NCL source unlike the NCL intrinsic function.

G

GIF
: A file format used for the storage and on-line retrieval of bitmapped graphical data. GIF stands for "Graphical Interchange Format"; it was created by the CompuServe Corporation in 1987.

graphcap
: A file that contains detailed information used to define the capabilities of a specific plotting device. Graphcaps have a human-readable ASCII form or a binary form that is readable by ctrans.

graphical object
: **NCL:** An NCL value of type graphic. A graphical object is an identifier for an HLU object.

Graphical User Interface
: A non-programmatic, graphical, interface to the functionality of NCAR Graphics. Such an interface is sometimes referred to as a "point-and-click" interface, since that is how the interaction is accomplished.

GUI
: Graphical User Interface.

H

HDF
: HDF (Hierarchical Data Format) is a multi-object file format, developed at NCSA, that facilitates the transfer of various types of data between machines and operating systems.

High Level Utilities
: Objects, like XyPlot objects, Contour objects, TextItem objects, and so forth, that can be created and manipulated by a set of library functions, callable from either a C program, a Fortran program, the NCAR Command Language, or a GUI. High Level Utilities are also referred to as HLUs and are to be distinguished from the Low Level Utilities, or LLUs.

HLU resource string
: A character string identifying a particular resource of a class. These are the resources listed in the descriptions of the classes.

HLUs
: High Level Utilities.

I

---

identifier
: **NCL:** A name of a variable, or a function, or a procedure.

immediate mode
: A mode used by certain API functions that produces immediate drawing without invoking a draw function.

instance
: When specific values are assigned to all the resources defined in a class, the result is an instance of that class. Any instance of any class is called an object. Default values exist for all resources; before creating an object, users may override any default value.

instance hierarchy
: Same as child/parent hierarchy.

inheritance
: A class is said to inherit resources or support functions from its superclasses, since those functions and resources are available to the subclass. See superclass, *and* composite class.

intrinsic annotation
: An annotation available as composite class member of the PlotManager class. Intrinsic annotations are available to any class of plot object and include TickMark, Title, LabelBar, and Legend annotations. The PlotManager manages these annotations internally.

intrinsic function
: **NCL:** An identifier with a list of parameters, the parameters being separated by commas enclosed and in parentheses. Intrinsic functions return values when called. An intrinsic function is not defined by NCL source; it is a C or Fortran routine that has been added to the NCL function set. Intrinsic functions often perform operations that NCL source does not support.

ISO
: The International Standards Organization that publishes international standards. (see ANSI).

intrinsic procedure
: **NCL:** An identifier with a list of parameters, the parameters being separated by commas enclosed in parentheses. An intrinsic procedure is not defined by NCL source; it is a C or Fortran routine that has been added to the NCL procedure set. Intrinsic procedures often perform operations that

NCL source does not support.

J

K

---

keyword
NCL: A word reserved by NCL that not allowed to be used as a variable or function name.

L

---

lazy evaluation
NCL: The process whereby relational expressions are assigned a value as soon as it is possible to do so, without necessarily evaluating all of the components in the expression. For example, the expression (1 .lt. 3) .or. (2 .lt. 1) can be assigned the value True immediately after evaluating (1 .lt. 3) without having to evaluate (2 .lt. 1).

literal array
NCL: An array of values specified using literal values, these values being separated by commas and enclosed in ' (/ ' and ' /) ' .

literal value
NCL: A single scalar value expressed by its actual string value (i.e. not referenced by a variable). For example, 1, 1.414 and "string" are literal values.

LLUs
Low Level Utilities.

local resources,
Resources defined in a particular class that are not inherited from another class.

Low Level Utilities
Traditional NCAR Graphics as it existed before Version 4.0. It is a package of about 500 graphics routines. User entries have both C and Fortran interfaces. These entries continue to be supported in 4.0.

M

---

marker
See polymarker.

member class
One of the class components of a composite class.

metadata

NCL: Information used to describe data, such as dimension names, variable attributes, valid ranges, and so forth.

metafile

A file containing encoded graphical elements. Metafiles are used for storing and transporting graphics images. In the context of NCAR Graphics "metafile" is generally synonymous with NCGM.

missing value

NCL: A special value for a variable or array element indicating that no legal data has been specified for that quantity. See the section on missing data in the evaluation of expressions in the NCL User Guide module on NCL expressions and operators for details on how these missing values are handled.

monotone

A sequence of numeric values is monotone (or monotonic) if either: each element in the sequence is larger than (or equal to) its predecessor, or each element in the sequence is smaller than (or equal to) its predecessor. A sequence is monotonically increasing if each element in the sequence is larger than its predecessor; a sequence is monotonically decreasing if each element in the sequence is smaller than its predecessor. A sequence is monotonically non-decreasing if each element in the sequence is larger than, or equal to, its predecessor. A sequence is monotonically non-increasing if each element in the sequence is smaller than, or equal to, its predecessor.

N

---

named dimension

NCL: A dimension of a variable or file variable that has been assigned a name using the '!' operator.

NCAR Command Language

A language written for the purpose of interactive data manipulation and display. NCL has a command line interface and will accept netCDF, HDF, Grib, or ASCII input files. NCL also provides an easy interface to the HLUs.

NCAR Computer Graphics Metafile

The NCAR private binary encoding of a Computer Graphics Metafile.

NCGM

NCAR Computer Graphics Metafile.

NCL

NCAR Command Language.

NCL resource list

NCL: An NCL resource list is a list of HLU resource strings followed by a " : " followed by a valid NCL expression.

NDC

Normalized Device Coordinates. A coordinate system that describes positions on a virtual plotting device. The lower left corner corresponds to (0,0), and the upper right corner corresponds to (1,1). NDC space will be mapped onto the largest square which will fit on an actual plotting device. PostScript output is centered on the page by default, but options exist for positioning PostScript output anywhere on the page. See the PSWorkstation for details.

netCDF
netCDF (network Common Data Form) is an interface for scientific data access and a library that provides an implementation of the interface.

numeric
NCL: Any data type that represents a numerical value. The numeric data types are: double, float, long, integer, short, and byte.

O

---

object
An object is created from a class by assigning specific values for the class resources. See class *and* instance.

overlay
A transform overlaid on a base plot using the add overlay function. The base plot sets the viewport of the overlay to match its own and transforms the coordinate data of the overlay into its own coordinate space. Only that portion of the overlay's coordinate space that intersects the coordinate space of the managing plot will be visible in the plot output. If the overlay is a plot object, it gives up its base plot status. The base plot to which the overlay is added assumes responsibity for managing the overlay's plot members.

overlay functions
The functions used to add and remove overlays to and from plot objects. Specifically, these functions are NhlAddOverlay and NhlRemoveOverlay for the C and Fortran interfaces.

overlay plot
An overlay created from a plot object rather than from a simple transform.

overlay sequence
The ordering of the transforms in a plot or subplot that contains overlays. The base plot is always first, followed by each overlay in an order that may be manipulated through the overlay functions. The overlay sequence determines the basic drawing order of the plot. The base plot is drawn first; each succeeding overlay is drawn on top of the preceeding transforms. Annotations are not affected by the overlay sequence; they always drawn after all the transforms.

P

---

parent
A relationship that exists between *objects*. If A and B are objects, then A is a parent of B if and only if B is a child of A. See child for more information.

polymarker
An array of coordinates specifying positions where certain specified symmetric symbols (markers) such as circles, dots, and so forth will be plotted.

plot
Depending on context, the word *plot* may be used to mean:

1. A plot object.
2. A primary base plot and all its plot members.

3. The output resulting from drawing a primary base plot and all its plot members.
4. The output resulting from drawing any arbitrary collection of viewable objects.

plot_class

NCL: The plot_class in NCL is the same as the HLU class pointer used in the HLU API to specify what type of object to create. The NCL plot_class identifier is spelled the same as the HLU class pointer.

plot member

A viewable object managed by a base plot. If the base plot is a subordinate base plot then the object is indirectly a plot member of the complete plot managed by the primary base plot. Drawing the primary base plot causes all its plot members to be drawn. A plot member is either an overlay or an annotation. Although an annotation can be any arbitrary view , an overlay must be a view belonging to the Transform class. A plot member must belong to the same Workstation as its base plot and cannot be drawn independently. A view cannot belong as a plot member to more than one base plot at a time.

plot object

A Transform object instantiated with an active PlotManager. In general, unless otherwise restricted by their specific class, plot objects have the ability both to manage other viewable objects as plot members and to be managed as plot members themselves. A plot object that manages plot members is called a base plot. If the managing plot object is itself managed as a plot member, it is a subordinate base plot. If it manages itself (i.e. is *not* a plot member) it is a primary base plot. At creation, all plot objects are primary base plots.

PostScript

A general-purpose programming language that contains a rich set of graphics operators. PostScript is produced by many popular word processing and graphics packages and can be displayed on a wide variety of printers, plotters, and workstation screens.

primary base plot

A self-managing plot object. A base plot that directly manages any number of plot members, consisting of overlays and annotations, but is not itself a plot member. At creation all plot objects are primary base plots. A plot object must be a primary base plot for the user to draw it or change its workstation.

procedure

NCL: An identifier with a list of parameters, these parameters being separated by commas and enclosed in parentheses. Procedures do not return values when called.

PS

see PostScript

Q

R

---

resource

A variable defined as part of the definition of a class. Resource values in *objects* can be set by using a Create function or a SetValues function. Resource values can be retrieved using a

GetValues function.

resource file
A file that can be used to set values for resources. There are four different resource files. See system resource file *and* user resource file.

resource forwarding
A technique that makes the resources of member classes available to a composite class.

RGB
Stands for the *red, green, blue color space* where colors are specified as triples of floating point numbers between 0. and 1. inclusive. The number triple gives the intensities for the red, green, and blue components of a color. The triple <1,.0.,0.> would indicate red, for example.

S

---

scalar
NCL: A single element of data of any type is referred to as a scalar value.

scalar_logical_expression
NCL: A single element value of the logical data type with no missing values.

scope
NCL: The range or area within a program in which an identifier is meaningful.

subordinate base plot
A plot member that is an annotation plot. A subordinate base plot sets the viewport of the plot members it controls, while its own viewport is set by the base plot that controls it. Unlike a primary base plot, the user cannot directly draw or change the workstation of a subordinate base plot. A subordinate base plot and the plot members it manages are known collectively as a subplot.

simple overlay
An overlay created from a simple transform.

simple transform
A transform object created without an active PlotManager. Unlike a plot object, a simple transform cannot manage any plot members, either as annotations or as overlays. Therefore, the only elements that appear when a simple transform is drawn are those implemented within the object itself. However, unless restricted by its particular class, a simple transform may itself become an annotation or an overlay of a base plot. Since a plot object has all the capabilities of a simple transform and none of the limitations, the main reason for creating a simple transform would be to conserve system resources when PlotManager capabilities are not required.

statement list
NCL: A sequence of statements separated by a carriage returns (\n).

statement
NCL: A single language construct within NCL that performs a specific task.

subclass
A class B is a subclass of A if B has in it all of the resources and support functions of A (B may have additional resources and support functions as well). If B is a class derived from A, then B is said to *inherit* its resources and support functions from A. If B is derived from A, then it is also said that B is a subclass of A.

subplot
The portion of a plot that is managed by a subordinate base plot.

superclass
A class A is a superclass of class B if A is on the same branch of the class hierarchy tree and A is higher on that branch.

superclass resource
A resource that one class inherits from a superclass.

support function
A function defined as part of a class definition.

system resource file
There are four files where resources can be set - two of these files are system resource files and the other two are user resource files. The name of one of the system resource files is specified by the setting of the environment variable NCARG_SYSRESFILE; the other system resource file is specified in an application-specific manner. See application resource file.

T

---

transform
A Transform object.

Transform class
The Transform class is a subclass of the View class that supports transformations from data coordinate space into the NDC space occupied by the Transform class instance's viewport. Transform subclasses may include the PlotManager class as a composite class member. The Transform class provides a resource for activating or deactivating the PlotManager when a Transform instance is created. In addition, the Transform class provides support functions for converting between data coordinate space and NDC space, for drawing immediate mode graphics primitives, and for adding and removing overlays and annotations from plot objects.

Transform object
An object that is an instance of the Transform class. Transform objects become plot objects when created with an active PlotManager instance. A Transform object created without an active PlotManager is called a simple transform.

U

---

user resource file
There are four files where resources can be set - two of these files are user resource files and the other two are system resource files. The name of one of the user resource files is specified by the setting of the environment variable NCARG_USRRESFILE; the other user resource file is specified in an application-specific manner. See application resource file.

V

---

variable
: **NCL:** A name that can contain a singly-dimensioned or multi-dimensioned data array, dimension names, coordinate variables, attributes, and so forth.

view
: A viewable object.

viewable object
: An object that is an instance of the View class.

View class
: An object can be drawn only if it is an instance of the View class. The View class provides resources for sizing and positioning objects on an output device (workstation). The View class also provides a support function for determining the bounding box of a given object in the class. A View class object must have a Workstation class parent.

View object
: A viewable object.

viewport
: For View objects, the viewport is a rectangular subregion of NDC space that specifies where the View object will be placed when drawn. The precise meaning of the viewport depends on the View object. For example, for XyPlot objects, the viewport specifies where the grid containing the curves will be placed, and the labeling (if any) will be drawn outside of the viewport. On the other hand, for TextItem objects, the viewport will be a rectangle surrounding the text string.

visualization block
: **NCL:** A group of NCL resources specified in either an NCL create, setvalues, or getvalues statement. Visualization blocks are used to create, modify, or inquire about the values of resources of objects.

W

---

workstation
: Used in NCAR Graphics Version 4 terminology to mean a valid output device such as an X Window System display, a PostScript file, or an NCGM.

workstation class
: A class that provides interfaces to specific output devices.

X

Y

Z

$Revision: 1.24 $ $Date: 1995/07/25 02:05:30 $

General troubleshooting

This troubleshooting module will be implemented in a later release of NCAR Graphics. For now, you might find help in the HLU troubleshooting module or the NCL troubleshooting module.

$Revision: 1.4 $ $Date: 1995/07/19 23:42:08 $