
Drawers

User Experience: Windows & Views



2003-02-04



Apple Inc.
© 2003 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Aqua, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY

DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction to Drawers 7

Organization of This Document 7

About Drawers 9

Positioning and Sizing a Drawer 11

Document Revision History 15

Figures

Positioning and Sizing a Drawer 11

- Figure 1 A drawer on the left edge of the parent window 11
- Figure 2 A drawer on the bottom edge of the parent window 12

Introduction to Drawers

A drawer is a special type of window that can slide out from one side of a window.

Organization of This Document

This document describes drawers and how they work.

The “Windows” chapter in *Aqua Human Interface Guidelines* discusses drawers from the point of view of how they relate to other Mac OS X user interface objects.

These are the concepts covered:

- [“About Drawers”](#) (page 9)
- [“Positioning and Sizing a Drawer”](#) (page 11)

About Drawers

`NSDrawer` is a user interface element that contains and displays view objects including `NSTextView`, `NSScrollView`, `NSBrowserView`, and other classes that inherit from `NSView`. A drawer is associated with a window, called its parent, and can only appear while its parent is visible on screen. A drawer cannot be moved or ordered independently of a window, but is instead attached to one edge of its parent and moves along with it. Drawers may be resized, but a drawer can never be larger than its parent. A given window may have any number of drawers; it is the developer's responsibility to make sure drawers do not overlap.

A drawer may be either open or closed, or in the process of opening or closing. When a drawer is closed, it does not appear on screen. A drawer may be opened on a particular edge of its parent depending on, for instance, the window's position on screen. Each drawer may also have a preferred edge which is used when an edge is not specified programmatically. If a closed drawer is told to open without a preferred edge specified, the drawer attempts to choose an edge on which to open based on the space available to display the drawer on screen. If you need to ensure that a drawer opens on a specific edge, use `openOnEdge:`.

The opening and closing of a drawer does not happen instantaneously, so a notification is sent whenever a drawer has finished opening or closing. A drawer may be told to open or close at any time, with the expectation that the command will eventually be carried out. If a rapid series of open and close commands is issued, the last one will override the others and will determine what state the drawer finally ends up in.

A drawer can only be associated with one parent window at a time, but the parent window can be changed at anytime using `setParentWindow:` (however, the change is only made when the drawer is closed). Each drawer has its own `keyViewLoop` separate from its parent.

Positioning and Sizing a Drawer

A drawer can be positioned on any edge of its parent window—left, right, bottom, or top. Figure 1 shows a drawer on the left edge of its parent, and Figure 2 shows a drawer on the bottom edge.

Figure 1 A drawer on the left edge of the parent window

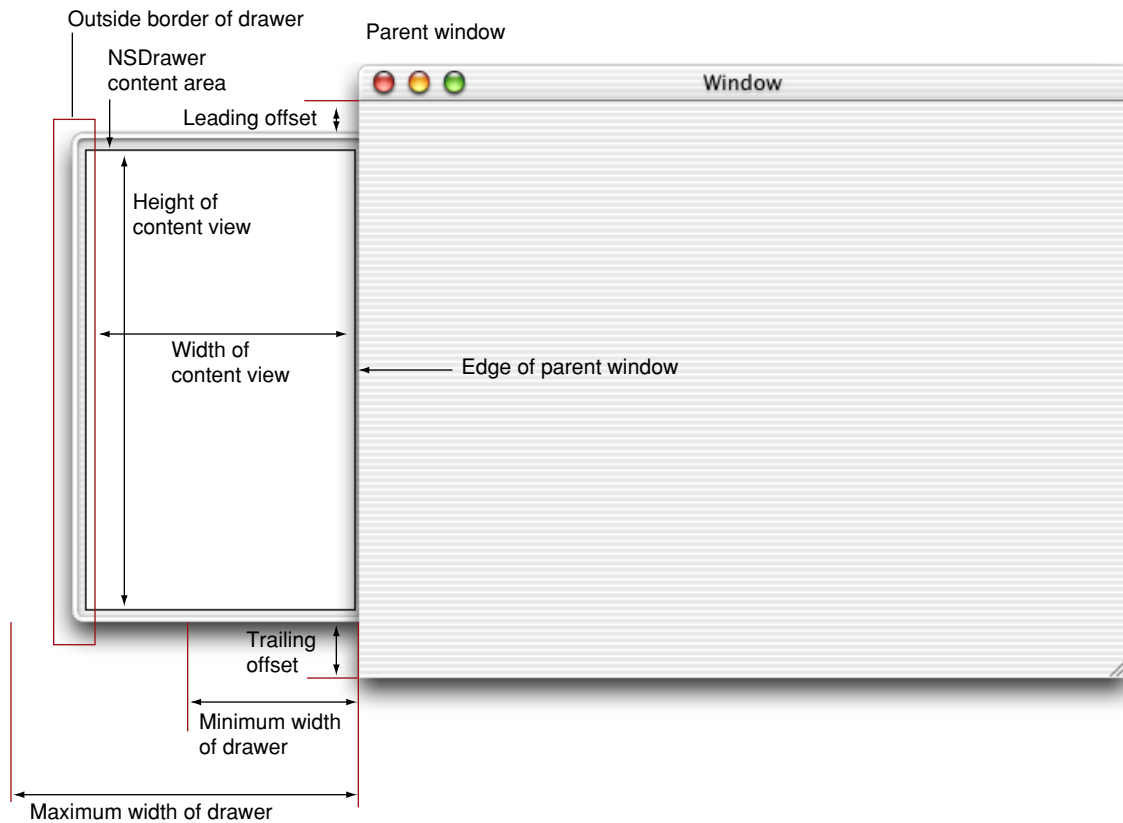
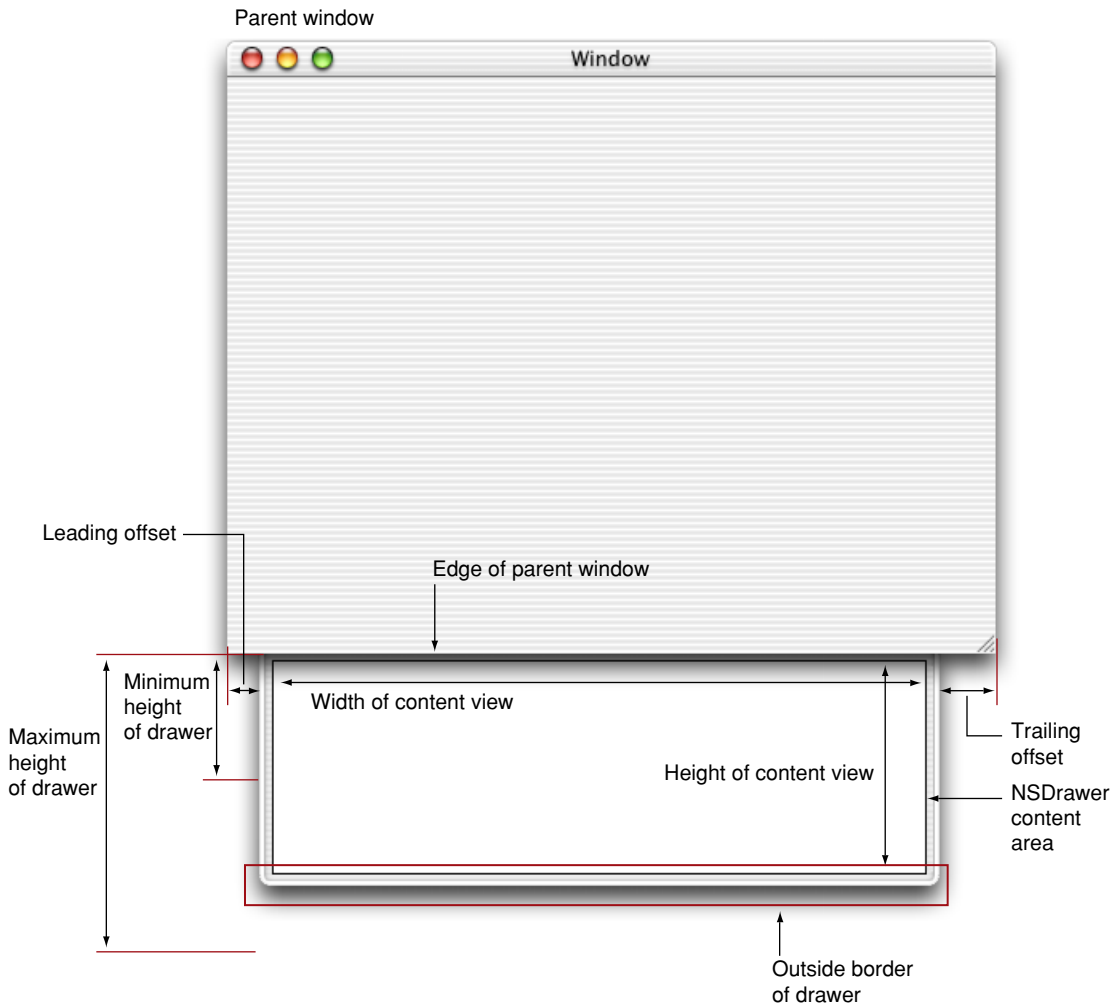


Figure 2 A drawer on the bottom edge of the parent window



A drawer has a leading offset and a trailing offset that can be set programmatically. The leading offset is the distance from the top or left edge of the parent window to the drawer and is set using `setLeadingOffset:`. The trailing offset is the distance to the right or bottom edge of the drawer from the right or bottom edge of the parent window and is set using `setTrailingOffset:`. The leading and trailing offsets for a drawer are returned using `leadingOffset` and `trailingOffset`, respectively.

The size of the content area of a drawer is set using `setContentSize:`. In the case of a drawer on the left or right edge of the parent window, the width of the content view is set according to the width specified in the content size. The height of the content view is a calculated value based on the leading and trailing offsets of the drawer and the height of the parent window, and is not affected by the height specified in the content size. In the case of a drawer on the top or bottom edge of the parent window, the height of the content view is set according to the height specified in the content size. The width of the content view is a calculated value based on the leading and trailing offsets for the drawer and the width of the parent window, and is not affected by the height specified in the content size.

A drawer cannot be larger than its parent window. An attempt to set the content size of the drawer to a value larger than the parent window is ignored and the content size is set to the maximum appropriate for the parent window.

The minimum size of a drawer is set using `setMinContentSize:.` In the case of a drawer on the left or right edge of the parent window, the width specified in the minimum content size is the minimum width the drawer can be resized by dragging its outside border without the drawer being caused to close. The height specified in the minimum content size is the minimum allowed height of the drawer when the parent window is resized. In the case of a drawer on the top or bottom edge of the parent window the height specified in the minimum content size is the minimum height the drawer can be resized by dragging its outside border without the drawer being caused to close. The width specified in the minimum content size is the minimum allowed width of the drawer when the parent window is resized.

If the settings of the leading offset, trailing offset, and the minimum content size of a drawer would cause the drawer to exceed the size of the parent window along `edge`, the window will automatically resize to fit the drawer. This can lead to confusing behavior for users if the new size of the window is greater than the currently set maximum size of the parent window.

The maximum size of the drawer content view is set using `setMaxContentSize:.` In the case of a drawer on the left or right edge of the parent window, the width specified in the maximum content size is the maximum width that the drawer can be resized by dragging its outside border. The height specified in the maximum content size is ignored. In the case of a drawer on the top or bottom edge of the parent window, the height specified in the maximum content size is the maximum height that the drawer can be resized by dragging its outside border. The width specified in the maximum content size is ignored.

The `DrawerMadness` example demonstrates several positioning and sizing patterns for drawers.

Document Revision History

This table describes the changes to *Drawers*.

Date	Notes
2003-02-04	Changed reference to the Aqua Human Interface Guidelines from “sheets” to “drawers”. Added html link.
2003-01-14	Added “Positioning and Sizing a Drawer” concept.
2002-11-12	Revision history was added to existing topic. It will be used to record changes to the content of the topic.

