Table of Contents

Acquired Knowledge, Inc.

Making Searchable PDF Files with Acrobat *Catalog*

PDF has become an extremely popular way to distribute documents on CD-ROM. Many newspapers make their back issues available as a set of CD-ROMs, each containing a year's worth of newspapers as PDF files. Surprisingly, many of these organizations fail to make use of one of the Acrobat package's most useful utilities: *Acrobat Catalog.*

What is it?

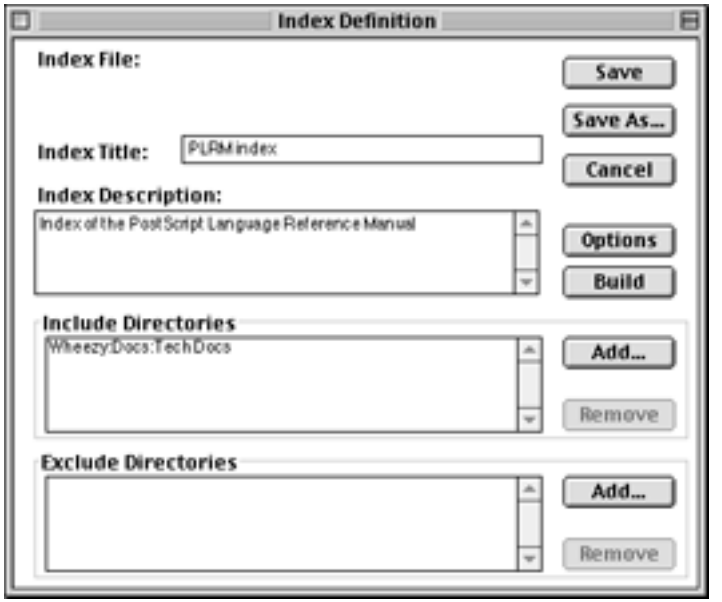
Acrobat Catalog is a utility that scans through a folder full of PDF files an makes a searchable index of the files' contents. When later viewing a PDF file in Adobe Acrobat, a user may click on the "search catalog" icon, specify a word or phrase, and be presented with links to all of the PDF files in the catalog that contained "hits," sorted by relevance.

The search is blindingly fast and it makes the set of PDF files making up a year's worth of newspaper issues far more useful than it would have been.

Next Page ->>

Creating a Catalog

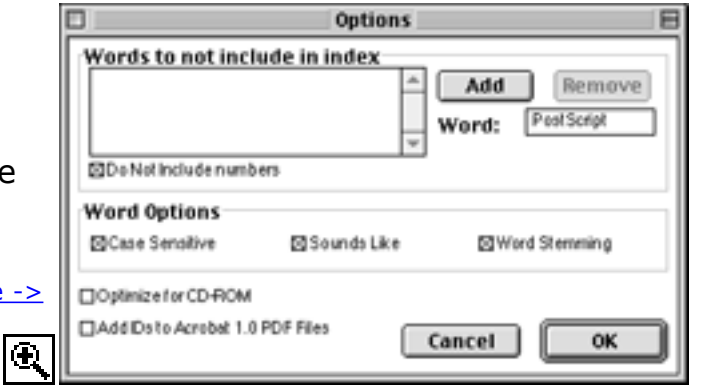
Having assembled your PDF files into a single directory (which will eventually be transferred to CD-ROM), you launch *Acrobat Catalog,* and select "New" from its File menu. *Catalog* will present you with it's mildly ugly "Index Definition" dialog box.



Give the new catalog a name descriptive of the document or set of documents to which the index belongs.

In the "Include Directories" section, click on the "Add" button and select the directory that contains your PDF files. *Catalog* will create an index from all PDF files in this directory and its descendants. If you want *Catalog* to ignore one of the descendent directories, you may tell it so by clicking the "Add" button in the "Exclude Directories" section.

Click on the "Options" button and you may specify a variety of parameters that affect Catalog's work, including a list of words it should ignore and whether searches should be case sensitive.



Next Page ->

Click the "Build" button, and Catalog will start working through the PDF files. The cataloging process takes a while, but when it is done, you will get an index file for the set of PDFs and a folder full of data files for the index, itself.

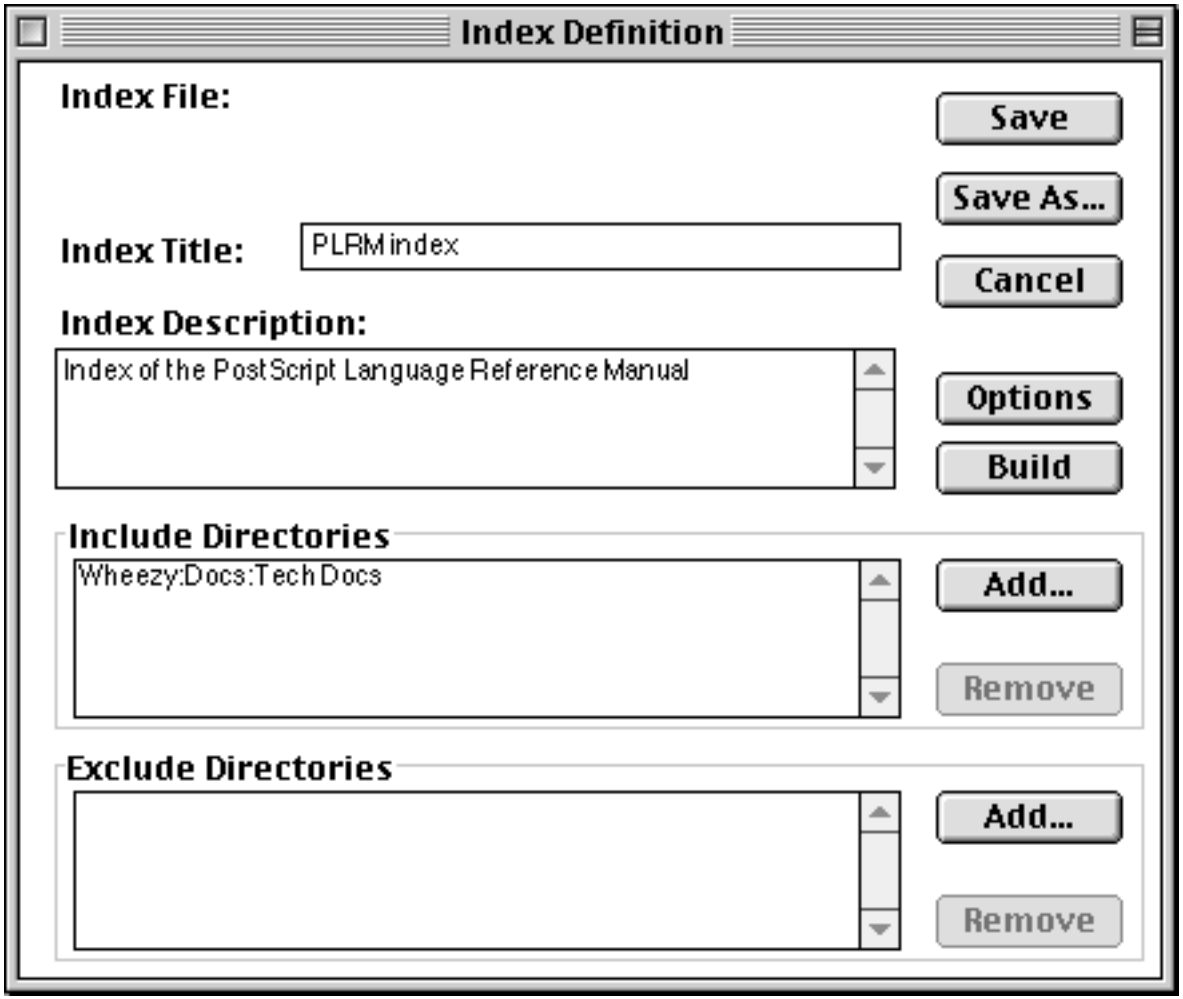
Place this index file and the data folder on the CD-ROM with your PDF files, and people using the documentation will be able to search back issues with ease.

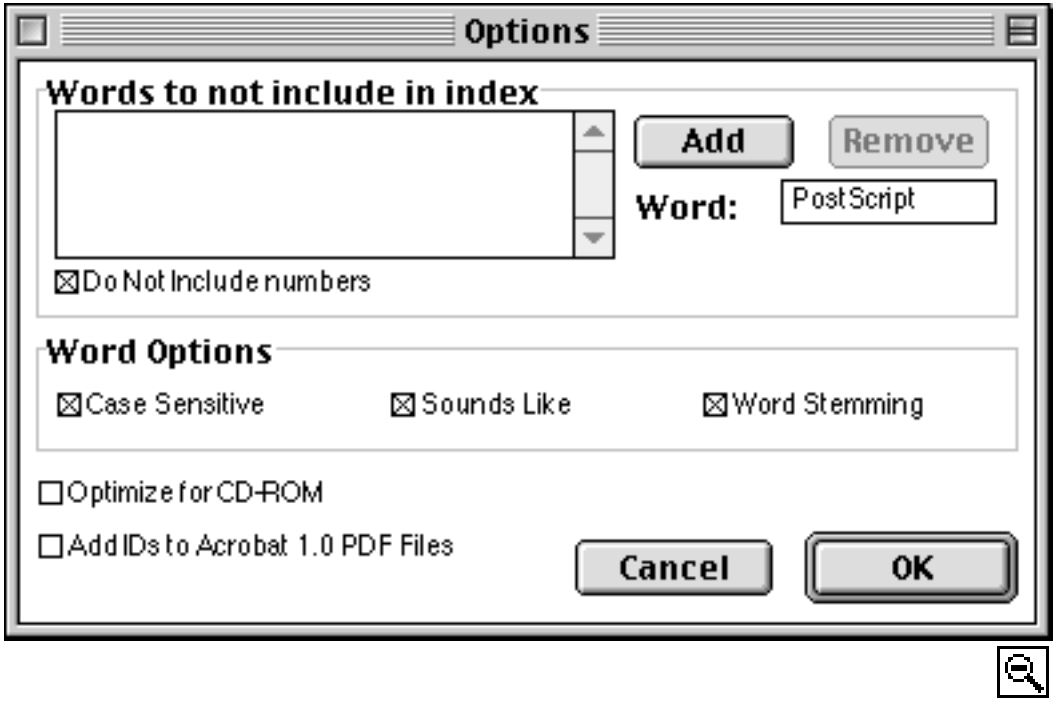
Tips

- If you move the indexed PDF files after creating the index, you must move the index file and data folder as well. The index file and folder must have the same relative position, compared to the files they pertain to, as when the index was created.
- For maximum compatibility, restrict your file names to the lowest common denominator: MS-DOS-style eight-dot-three naming. If this is too restrictive (it is!), then at least avoid names containing characters that are special in other computers' file systems; in particular, file names should not contain colons, slashes, or backslashes.
- Indices can be very large, up to 40% of the size of the PDF files it catalogs. Be sure to reserve space on your CD-ROM for the index and its folder.

This article is only a brief introduction to *Catalog.* Be sure to read your Acrobat on-line reference for a full in-depth description of how to use this extremely useful utility. A good, searchable index adds immeasurably to the usefulness of an on-line document.

Newsletter First Page


Index Definition
Index File:
Save
Save As...
Index Title:
PLRMindex
Cancel
Index Description:
Index of the PostScript Language Reference Manual
Options
Build
Include Directories
Wheezy:Docs:Tech Docs
Add...
Remove
Exclude Directories
Add...
Remove


Options
Words to not include in index
Add
Remove
Word:
PostScript
Do Not Include numbers
Word Options
Case Sensitive
Sounds Like
Word Stemming
Optimize for CD-ROM
Add IDs to Acrobat 1.0 PDF Files
Cancel
OK

A Taste of Smooth Shading

This month, I thought I'd present a very brief introduction to one of my favorite parts of PostScript LanguageLevel 3: Smooth Shading.

If you aren't familiar with it, smooth shading provides the PostScript language's support for gradients, a smooth transition from one color to another. We are going to look at the LanguageLevel 3 code that produces the following gradient-filled rectangle.

Our discussion of this PostScript code will be necessarily abbreviated, but it should be more than enough to let you modify the PostScript code to produce your own gradients.

Next Page ->

What's So Special?

What's special about LanguageLevel 3's smooth shading is that the gradients it produces are calculated at the machine level. They are fast and stunningly good.

Since both PostScript LanguageLevel 3 and PDF 1.3 support smooth shading, you can get a feel for what I mean by zooming in on the gradient example below. Zoom in to 1600% and you will see how smooth the gradient remains.

(You do have your monitor set to zillions of colors, don't you?)

Next Page ->

Without Smooth Shading

Before we proceed, consider how you'd produce the our smooth shaded image without smooth shading.

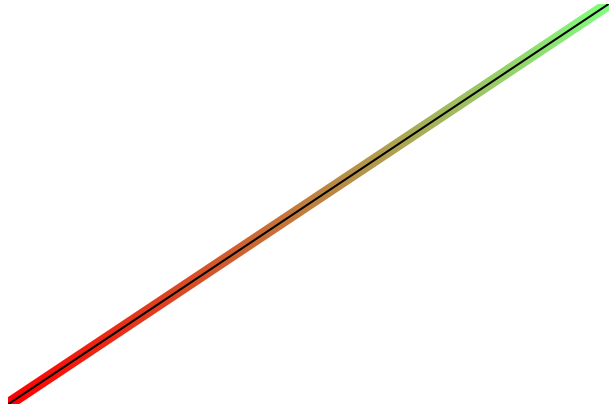
You would have to either print a whole lot of skinny rectangles of varying color going from one end of the gradient to the other (inelegant and grainy) or you'd have to construct an image with the appropriately varying color data and print that (better results, but still grainy and can require a very large amount of data).

As we'll see, the smooth shading code that produces this is simple, elegant, and produces absolutely smooth results.

Next Page ->

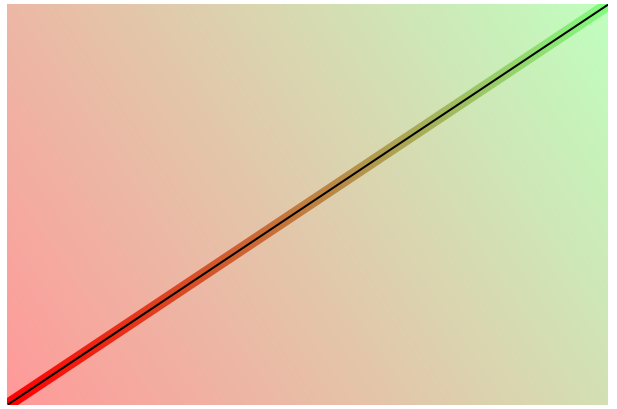
Overview of the Process

In LanguageLevel 3, the gradient we are using is called a **Type 2** smooth shading; the color varies along an axis that extends, in this case, from the lower-left to the upper-right corner of the rectangular area



To produce this shading, we need to supply two broad pieces of information:

- The beginning and end points of the axis along which the color will vary.
- The exact way the color will vary as we go from one end to the other of the axis. In our case, we'll be supplying the color at the end-points and letting PostScript do an interpolation along the axis.



The gradient we create will extend perpendicular to our axis out to the boundary of the clipping path.

Let’s see how we do this.

Next Page ->

The PostScript Code

Here's the PostScript code that produces the above gradient-filled rectangle:

```
100 400 300 200 rectclip

<<    /ShadingType 2
      /ColorSpace [ /DeviceRGB ]
      /Coords [100 400 400 600 ]
      /Function <<
            /FunctionType 2
            /Domain [ 0 1 ]
            /C0 [ 1 0 0 ]
            /C1 [ .5 1 .5 ]
            /N 1
      >>
>>
shfill

showpage
```

The program starts with a call to *rectclip,* which limits the gradient we shall create to the specified rectangular area.

This rest of the program consists of a call to the LanguageLevel 3 *shfill* operator. This operator takes a dictionary from the stack (a **shading dictionary),** and uses the information contained in that dictionary to create the gradient.

Next Page ->

Shading Dictionaries

Within the shading dictionary, we find four key-value pairs:

```
/ShadingType 2
```

This tells PostScript what kind of smooth shading you want. Here we are saying we want Axial Shading.

```
/ColorSpace [/DeviceRGB ]
```

This tells PostScript that we shall be specifying colors in RGB.

```
/Coords [ 100 400 400 600 ]
```

This defines the endpoints of our axis. The four numbers are the x,y coordinates of the beginning and end of the axis.

```
/Function <<...>>
```

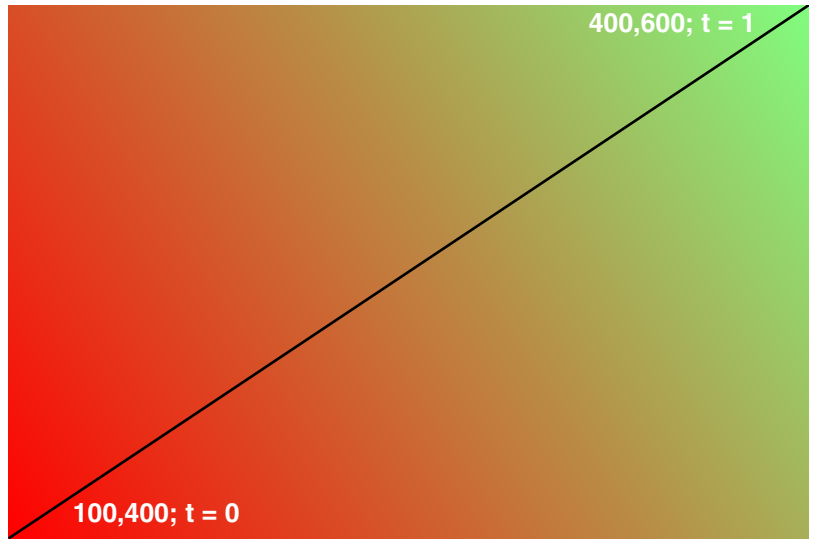
This is a PostScript **Function Dictionary,** a construct introduced in LanguageLevel 3. The function dictionary we are using defines how color will vary as we move from one end to the other of the axis. (We'll talk about it's contents on the next page.)

Next Page ->

AKI Journal: PostScript Tech

Function Dictionaries

Our function dictionary specifies the colors at the two endpoints of the axis and then interpolates between those two colors as we move from one end to the other. In effect, there is an internal parameter, *t,* that varies from 0 to 1 as we move from one endpoint to the other. The function dictionary calculates color, RGB, as a function of *t.*



The key-value pairs we use to specify this relationship are:

`/FunctionType 2`

This tells PostScript that the function we want to use interpolates between the two endpoint colors. PostScript recognizes three different types of function dictionary.

`/Domain [ 0 1 ]`

This specifies the set of values across which *t* will vary as we go from one endpoint to the other. Here, *t* will vary from 0 to 1.

Next Page ->

```
/C0 [ 1 0 0 ]
```

This array contains the RGB value for the color at endpoint 0.

```
/C1 [ .5 1 .5 ]
```

This array contains the RGB value for the color at endpoint 1.

```
/N 1
```

This number is the exponent for the interpolation. The 1 specifies we want a linear interpolation between the endpoints. (2 would be quadratic, 3, cubic, etc.)

Next Page ->

AKI Journal: PostScript Tech

That's It!

That's an example of how to do a gradient in PostScript LanguageLevel 3. Not too hard, eh?

What we haven't discussed here: other shading types (there are seven of these, including radial shading and various types of meshes) and other types of function dictionaries.

For More Information

If you want to learn more about Smooth Shading, see the appropriate section in the PostScript Language Reference Manual. You can, of course, also take the Acquired Knowledge *Advanced PostScript* course, which will tell you all about Smooth Shading, as well as forms, filters, binary encoding, and everything else that goes into LanguageLevels 2 and 3.

Click here if you want to go to Acquired Knowledge's web site for information about the Advanced PostScript class.

Newsletter First Page

Schedule of Classes, Nov 2000 - Jan 2001

Following are the dates and locations of Acquired Knowledge's PostScript and Acrobat classes. Clicking on a class name below will take you to the Acquired Knowledge web-site to the description of that class.

PostScript Classes

PostScript Foundations	San Diego, CA	Nov 13 - 17
Advanced PostScript	Rochester, NY	Nov 6 - 9
	San Diego, CA	Dec 11 - 14
PS for Support Engineers	San Diego, CA	Nov 6 - 10
	London, UK	Nov 13 - 17
	Seattle, WA	Dec 4 - 8

PostScript Course Fees PostScript classes cost $1,900 per student, except for courses conducted in the U.K., which cost $2,000 per student.

Acrobat Classes ->

Registration ->

Acrobat Classes

Acrobat Essentials & Interactive Acrobat

Detroit, MI	Nov 1	San Francisco, CA	Dec 11
Grand Rapids, MI	Nov 2	Los Angeles, CA	Dec 13
New York, NY	Nov 7	Orange County, CA	Dec 15
Philadelphia	Nov 9	Phoenix, AZ	Jan 9, 2001
Washington, DC	Nov 14	Denver, CO	Jan 11
Rochester, NY	Nov 16	Chicago, IL	Jan 16
Houston, TX	Dec 4	Minneapolis, MN	Jan 18
San Antonio, TX	Dec 6	Newark, NJ	Jan 23
San Diego, CA	Dec 8	Long Island, NY	Jan 25

Acrobat Course Fees

The Acrobat Essentials and Interactive Acrobat classes each cost $195.00 per student. You may take both for $350.00. There is a discount if you are signing up 3 or more people.

<- Back to PostScript Classes

Registration ->

Contacting Acquired Knowledge

For more information

For class descriptions or for any other information about AKI's classes or software:

Web site: http://acquiredknowledge.com

email: info@acquiredknowledge.com

telephone: 800-482-1252

mail: 6480 Weathers Place #245, San Diego, CA 92121

Registering for Classes

To register for an Acquired Knowledge class contact us any of the following ways:

Register On-line: http://www.acquiredknowledge.com/courses

email: registration@acquiredknowledge.com

telephone: 800-482-1252

mail: 6480 Weathers Place #245, San Diego, CA 92121

Newsletter First Page

What's New at Acquired Knowledge?

EZ-PDF 2.0 Ships!

EZ-PDF 2.0 for Macintosh and Windows is now shipping. This new version adds complete support for all Acrobat 4.0 features, as well as giving access to many of Acrobat's "hidden" options.

To download a full-featured, 30-day demo go to:

http://www.acquiredknowledge.com/downloads.html

New PitStop class

Acquired Knowledge is now teaching *Preflight and Troubleshoot with Enfocus' PitStop.* This one-day seminar teaches the student how to use Enfocus Software's popular *PitStop* software. This class teaches you *everything:* how to set up preflight profiles; how to edit text and line art; how to apply global changes to color, text, and other contents of your Acrobat document, and more!

For more information, go to

http://www.acquiredknowledge.com/Courses/pitstop.html.

Newsletter First Page

Journal Feedback

If you have any comments regarding the AKI Journal, please let us know. In particular, we are looking for three types of information:

Comments on usefulness. Does the Journal provide you with worthwhile information? Was it well written and understandable? Did you like it, hate it, or did it fill you with ennui? How could we make it better? Do you like the PDF format?

Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like us to address?

Questions and Answers. We are planning a Q&A section for future issues. Do you have any questions about Acrobat, PDF or PostScript?

Please send any comments, questions, or problems to

journal@acquiredknowledge.com

Newsletter First Page