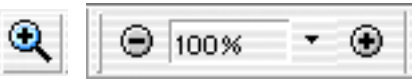
Table of Contents

Acumen Training

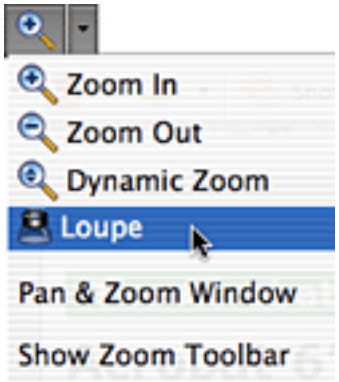
This month's articles (and next's) are relatively short, since John's up to his ears working on the *PDF File Content and Structure* class.

Acrobat 6's Zoom Tools

The ability to zoom in on to a document to see its details or to zoom out to get an overall view is extremely important any document exchange and viewing software, including Acrobat. In Acrobat 5 and earlier, we had access to a set of magnifying flasses and a toolbar control that let us specify a magnification percentage.

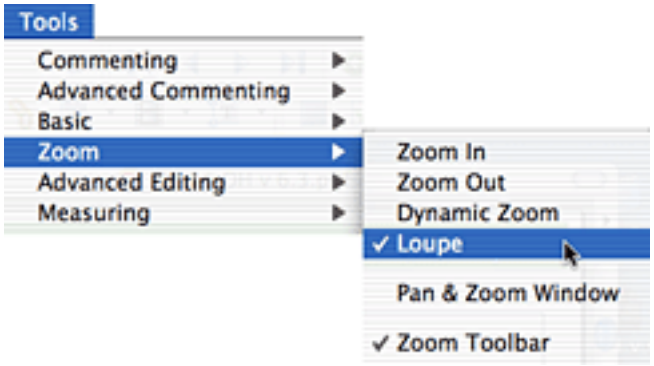


Acrobat 6 adds some additional methods for zooming into and out of a document:



- A *Loupe* tool that lets you zoom in on specific points on the Acrobat page.
- A *Pan & Zoom Window* that lets you zoom in on a particular area on the page.
- A *Dynamic Zoom* tool that lets you zoom in and out of the page in a variable, on-the-fly manner.

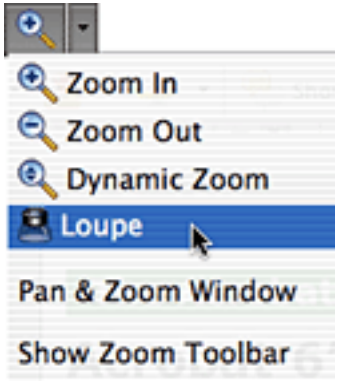
These tools are available from the *Zoom* toolbar, as above, or from the *Tools* menu, as at right.



This month's short article will examine each of these in turn.

Next Page ->

The Loupe Tool



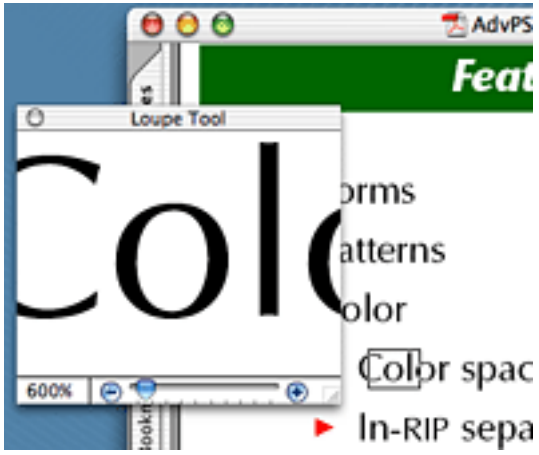
We'll start with the Loupe tool simply because it's the one I use the most.

This tool is similar to many tools available on the Macintosh and Windows for general use. It lets you zoom in on a particular point on the Acrobat page.

When you select the tool, the cursor turns to a crosshair with a box, as at left. When you click on the page, Acrobat displays a floating window that shows a close-up of the page centered on the point where you clicked. A rectangle on the page shows the area visible in the close-up.

As is true of all the Acrobat zoom tools, the page is active in the background while the Loupe tool is selected. You can click on the Next page button and otherwise navigate around the document as usual.

The slider in the floating window lets you select a magnification, as at right.

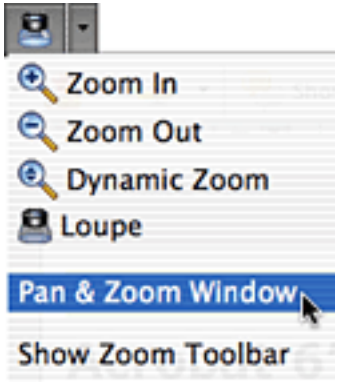


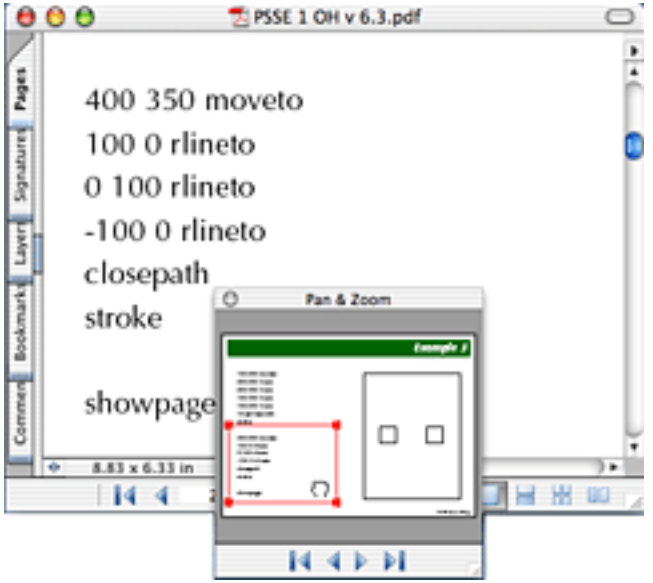
As long as the Loupe tool is selected, you can drag the "active" rectangle where you wish on the page.

Next Page ->

Pan & Zoom Window

The Pan & Zoom window presents you with a thumbnail view of the current page upon which is superimposed a red "selection rectangle" with handles in each corner.



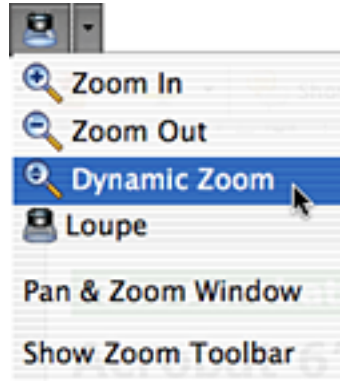


You can drag this rectangle around the page and resize it by grabbing the handles. As you do so, the image in the Acrobat document window, in the background, will display the contents enclosed by the selection rectangle.

Note that the Pan & Zoom window has navigation controls running along the bottom, beneath the thumbnail image. These mirror the functions of the equivalent controls in the Acrobat toolbar. Both sets of navigation controls—those in the Pan & Zoom window and in the toolbar—are active.

Next Page ->

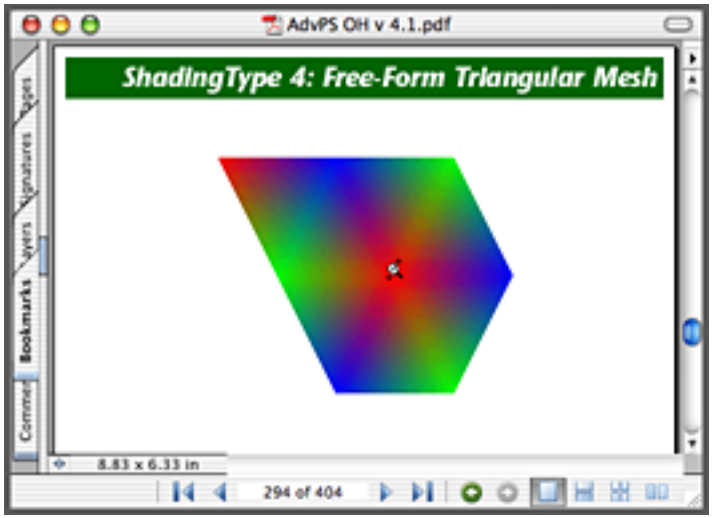
Dynamic Zoom Tool

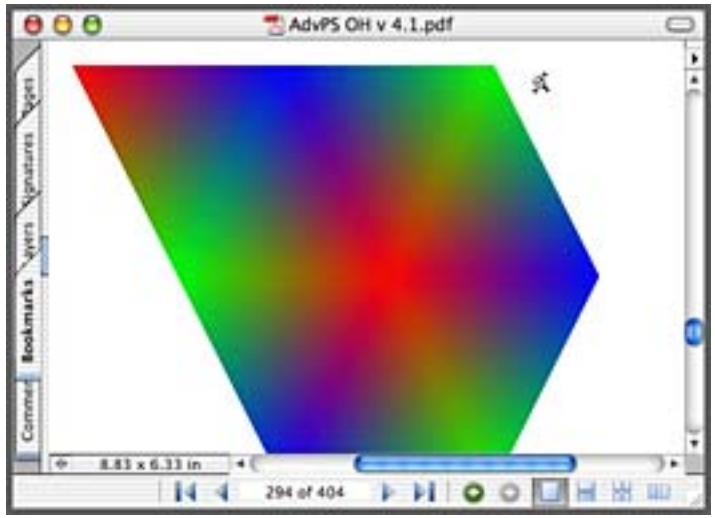


Finally, the Dynamic Zoom tool lets you smoothly zoom in or out, selecting a magnification by eye.

When you select this tool, the cursor turns to a little hard-to-describe icon shown at left.

When you click on the Acrobat page and then drag the cursor, Acrobat smoothly changes the magnification, zooming in if you drag the cursor upward, zooming out if you drag downward. The point where you first clicked remains fixed in the Acrobat window.





The Acrobat 6 Zoom tools are a very good addition to the Acrobat interface. Certainly I find myself using the Loupe and Pan & Zoom tools quite a lot.

Return to Main Menu

Transfer Functions

One of my favorite parts of PostScript, when I'm in certain rare moods, is the transfer function. This is a procedure body, residing in the graphics state, that specifies how the current shade of gray should be modified to accommodate the printing characteristics of the current device.

Adjusting Gray Values

Some devices intrinsically print darker than others; if you ask for a `.5 setgray`, the output from a particular device may look darker or lighter than a medium gray. On some devices, if you want a 50% gray, you should actually specify, say, `.6 setgray`; on a particular too-dark printer, this would look like a medium gray.

The PostScript transfer function is the mechanism by which PostScript converts the value you pass to *setgray* into the gray value that produces the desired tint on the page. It is a procedure that expects a gray value from the stack and returns the gray value that should be used instead. If your PostScript code specifies `.5 setgray`, *stroke, fill,* and the other painting operators push the current gray value on the operand stack and execute the transfer function, which must convert the *.5* into whatever gray value (*.6,* perhaps) looks like a 50% gray on the particular device.

Part of doing a PostScript printer implementation is writing a default transfer function appropriate to that printer's marking engine.

A PostScript program can change the transfer function withinin a page description to achieve a variety of special effects, including negative printing and "posterization." In this article we shall see how to do this.

Next Page ->

Acumen Journal: PostScript Tech

The *settransfer* operator

The PostScript *settransfer* operator takes a procedure body from the operand stack and makes that procedure the current transfer function.

```
{ xfer } settransfer => ---
```

From this point, on, every painting operation (*show, stroke, fill, image,* etc.) will use the new transfer function.

Let's look at some examples of how to use this to produce some printing effects applied to the image at right.

This image was originally exported from Photoshop as an EPS file. To affect the transfer function used by this image, I am pasting my call to *settransfer* immediately after the setup section of the PostScript.

```
...
...
%%EndSetup
{ ... } settransfer
```

Next Page ->

Negative print

These images reside in the file *settransfer.zip* on the Acumen Training *resources* page.

Let's start with converting our image to a negative, as at right. We shall need a transfer function that causes dark and light gray to exchange places; black becomes white, white becomes black, etc.

The easiect way to do this is to subtract the current gray value from 1; if the current gray is *g,* our new gray value will be *1 – g.*

Here's the transfer function that does the job:

```
{ 1 exch sub } settransfer
```

Remember that this procedure will receive the current gray value as an argument passed on the operand stack. The procedure places a *1* on the stack, reverses the two numbers with *exch,* and then subtracts the two values.

The resulting gray value, left on the stack, will be used to paint the page. It will be the optical inverse of the original gray value.

Next Page ->

Turn up the Contrast

Consider the following transfer function:

```
{ .5 gt { 1 } { 0 } ifelse } settransfer
```

This transfer function examines the shade-of-gray argument and replaces it with 1 (white) if the value is light gray (*i.e.,* greater than .5) and with black if the gray value is less than .5.

This results in turning the contrast all the way up within the image; all points in the image become either black or white, all grayscale values being moved to one extreme or the other.

Next Page ->

Posterization

As our final example, we shall "posterize" our image, using a transfer function that converts all of the shades of gray to one of 11 values: 0, .01, .02, ..., 1.0.

We could do this with a set of nested *ifelse* clauses:

```
{     dup .1 lt              % Is currentgray < .1?
        { pop 0 }            % Yes: substitute 0
        { dup .2 lt          % else, is gray < .2?
           { pop .1 }        % Yes: substitute .1
           { ...repeat until tired  % etc.
           } ifelse
        } ifelse
} settransfer
```

This is tedious, however, and I think the code is esthetically unpleasing.

Let's do it this way, instead:

```
{ 10 mul round 10 div } settransfer
```

This multiplies our gray value by *10,* yielding a floating point value between 0 and 10. We then round this number off to an integer and then divide it by *10*. the result is a number from *0* to *1* that will be one of the set [0, .1, .2, .3, ... 1.0].

Next Page ->

This Affects Everything

Keep in mind that the transfer function affects *all* painting. In this article, we are applying it to an image. However, when you specify a new transfer function, it will affect literally everything you paint on the page.

Concatenating Transfer Functions

Our calls to *settransfer* are completely replacing the current transfer function with one of our own. This is a problem because we completely discard the default transfer function, created by the printer manufacturere to allow for the PostScript device's particular printing characteristics.

In order to preserve the device's default grayscale adjustment, we need to concatenate our transfer function to the currently-existing function. This pretty easy if you remember two facts about Postscript arrays:

[and] are operators

- The [open- and close-brackets] that we normally use to create arrays are actually a pair of PostScript operators.

  The open bracket puts a mark object on the stack and that is *all* it does. PostScript code following the open bracket executes as usual, accumulating objects on the stack.

  The close bracket operator collects everything off the stack through the first mark it finds and constructs an array from those objects.

Next Page ->

Thus, the following PostScript line

```
[ 1 2 3 ]
```

first places a mark object and then the numbers *1, 2,* and *3* on the stack; the close bracket then constructs an array from those objects, leaving that array on the stack.

Procedures are arrays

- Procedure bodies, such as our transfer functions, are actually executable arrays; you can do anything to a procedure body that you can do to any other array.

Concatenating Functions

Given the above two facts, here is how we would concatenate our "negative" function with the current transfer function:

```
[                          % Put a mark on the stack
     currenttransfer       % Put the current transfer function on the stack
     aload pop             % Unload the function
     { 1 exch sub }        % Put our function on the stack
     aload pop             % Unload it
]                          % Create the array from the combined fcn. contents
cvx                        % Convert it to executable (i.e. to a proc)
settransfer                % Make the concatenated fcns our new xfer fcn
```

This will cause our PostScript file to print as a negative, while still preserving whatever device-specific adjustment the printer manufacturer has provided.

Let's look at this step-by-step.

Next Page ->

Acumen Journal: PostScript Tech 12

Step-by-step

```
[                          % stack: mark
```

We start with an open bracket, which puts a mark object on the stack. This marks the beginning of our array definiation.

```
currenttransfer            % mark { xfer }
```

The *currenttransfer* operator leaves the current transfer function procedure on the operand stack.

```
aload pop                  % mark obj0 obj1 ...
```

The *aload* operator takes an array as its argument (in our case, the procedure body for the current transfer function) and "unpacks" it, placing its contents, in order, on the operand stack, followed by a copy of the array, itself. We *pop* the returned copy of the procedure body, leaving on the operand stack all of the the objects contained within the original procedure body.

```
{ 1 exch sub }             % mark obj0 obj1 ... { proc }
```

We place our "negative" transfer function procedure on the stack…

```
aload pop                  % mark obj0 obj1 ... 1 -exch- -pop-
```

…and unload its contents onto the stack.

Next Page ->

```
]                                % [array]
```

We use the close bracket operator to pack the objects on the operand stack (down through the mark, anyway) into an array. This array contains all the objects from the current transfer function followed by the objects that make up our new procedure body. It is a concatenation of the two executable arrays.

This is a literal array, of course, not yet a procedure.

```
cvx                              % {proc}
```

We use *cvx* to convert the array into an executable array, that is, a procedure body.

```
settransfer
```

Finally, we set the current transfer function to our concatenated procedure body.

All future painting onto the page will use a shade of gray that is first adjusted for the current device and then subtracted from one.

This is the proper way to use the transfer function to implement special effects.

Return to Main Menu

Schedule of Classes, Sept – Nov 2003

Note that there are no classes in August. John's busy working on the *PDF File Contents and Structure* class.

Following are the dates and locations of Acumen Training's upcoming PostScript and PDF Technical classes. Clicking on a class name below will take you to the description of that class on the Acumen training website. The Acrobat class schedule is on the next page.

The PostScript classes are taught in Orange County, California and on corporate sites world-wide. See the Acumen Training web site for more information.

Technical Classes

New!	**PDF File Content and Structure**	Sep 29–Oct 2		
	PostScript Foundations	Sep 15–19		Nov 3–7
	Variable Data PostScript		Oct 20–24	
	Advanced PostScript			Nov 17–21
	PostScript for Support Engineers			
	Jaws Development		*On-site only*	

On-site Classes

These classes may also be taught on your organization's site. Go to Acumen Training's On-site Classes page for more information.

Technical Course Fees The PostScript and PDF classes cost $2,000 per student.

Registration Info →

Acrobat Classes →

Acrobat Class Schedule

These classes are taught quarterly in Costa Mesa, California, and on corporate sites. Clicking on a course name below will take you to the class description on the Acumen Training web site.

Acrobat Essentials — *No Acrobat classes scheduled for this quarter.* See the Acumen Training website regarding setting up an on-site class.

Interactive Acrobat

Creating Acrobat Forms

Troubleshooting with Enfocus' PitStop

Acrobat Class Fees — *Acrobat Essentials* and *Creating Acrobat Forms* ($\frac{1}{2}$-day each) cost $180.00 or $340.00 for both classes. *Troubleshooting With PitStop* (full day) is $340.00. In all cases, there is a 10% discount if three or more people from the same organization sign up for the same class.

Registration ->

Return to Main Menu

Contacting John Deubert at Acumen Training

For more information

For class descriptions, on-site arrangements or any other information about Acumen's classes:

Web site: http://www.acumentraining.com **email:** john@acumentraining.com

telephone: 949-248-1241

mail: 25142 Danalaurel, Dana Point, CA 92629

Registering for Classes

To register for an Acumen Training class, contact John any of the following ways:

Register On-line: http://www.acumentraining.com/registration.html

email: registration@acumentraining.com

telephone: 949-248-1241

mail: 25142 Danalaurel, Dana Point, CA 92629

Back issues

Back issues of the Acumen Journal are available at the Acumen Training website: www.acumenjournal.com/AcumenJournal.html

Return to First Page

What's New at Acumen Training?

Acumen Training is moving

Acumen Training will be moving at the end of August. As of September 1, 2003, the new address will be

24996 Danamaple
Dana Point, CA 92629

Telephone number and email address will be unchanged.

PDF Class Still on Schedule

Not news, exactly, but for those who have asked: the *PDF File Content and Structure* class is still on schedule for its first presentation the week of September 29. The course description and schedule are available on the Acumen Training website. The class will be available for teaching on-site beginning in October.

Return to First Page

Journal Feedback

If you have any comments regarding the *Acumen Journal,* please let me know. In particular, I am looking for three types of information:

Comments on usefulness. Does the Journal provide you with worthwhile information? Was it well written and understandable? Do you like it, hate it? Does it make you think that perhaps Frodo Baggins was right?

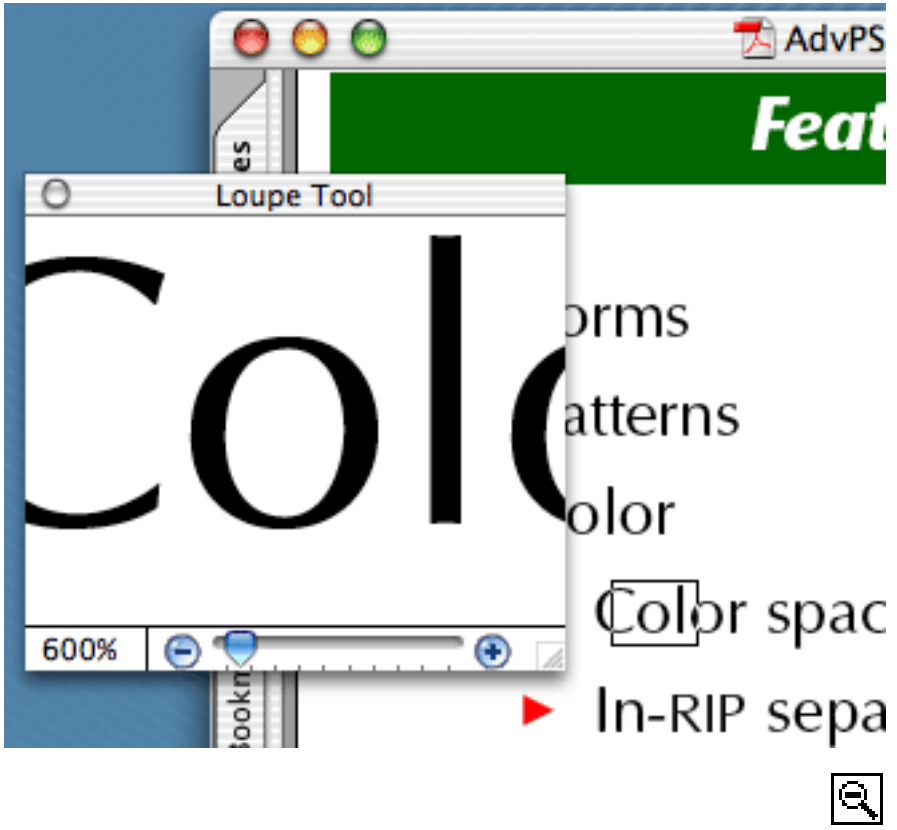
Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like me to write about?

Questions and Answers. Do you have any questions about Acrobat, PDF or PostScript? Feel free to email me about. I'll answer your question if I can. (If enough people ask the same question, I can turn it into a Journal article.)

Please send any comments, questions, or problems to:

journal@acumentraining.com

Return to Menu


AdvPS
Feat
Loupe Tool
Colo
600%
orms
atterns
olor
Color spac
In-RIP sepa