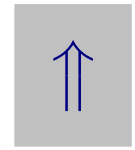
THE UNIVERSITY OF AKRON
Mathematics and Computer Science



mainmenu

Article: Links and Forms using Pdfmarks

Directory

Last Revision Date: 4/7/2001

Typeset by $\mathcal{AMS}$-TEX
& The Y&Y TEX System

Links and Forms using Pdfmarks

Table of Contents

1. Hypertext Links

- **Introduction**

When constructing `pdf` files, additional features unique to an electronic presentation can be included in the document by using Adobe's Acrobat application (formerly known as Exchange. This is a very powerful and useful program but it requires the author of the document to create these special features "by hand." Within the context of a web site that is constantly undergoing changes and revisions, it is necessary to have a way of introducing these features automatically into the document without any significant author intervention. This is the roll the `pdfmark` plays.

In this article, I survey some of the many `pdfmarks` that I have found useful in the process of preparing the two mathematical tutorials, e-Calculus and Algebra Review in Ten Lessons, and the mathematical games, Algeboard and Giants of Calculus.

This article is not a complete survey of `pdfmarks`, actually, one can think of this article as a *supplement* to Thomas Merz's The Pdfmark Primer, available over the Web.

Speaking of Thomas Merz, if you're thinking of putting `pdf` files on the WEB, may I recommend Thomas' new book Web Publishing with Acrobat/PDF ([webpub]). This book is a very fine and comprehensive resource of information. (Chapter 6 is The Pdfmark Primer referenced above). A Table of Contents has been made available for your inspection.

The ultimate resources for `pdfmarks` are Adobe's technical publications:

- **pdfmark Reference Manual**,
  Technical Note #5150
- **Portable Document Format Reference Manual**,
  Version 1.2

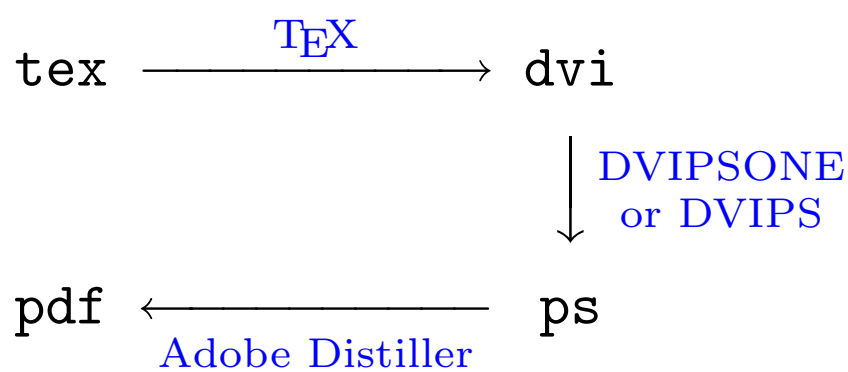
Both of these documents come with the Acrobat software, or they are available from the Adobe Web site, see Other Technical Notes.

There are many, many topics not covered in these notes such as notes, bookmarks, article threads, page cropping, etc. Should you be interested in topics beyond the set bounds of these articles, there is always the above mentioned Adobe Publications, or The Pdfmark Primer.

Forms and Links afforded me the opportunity to finally organize my knowledge and experiences of `pdfmarks`. Should you find these notes useful, that would be all for the good. 𝔇𝔖

1.1. AcroTEX

AcroTEX[1] is a term my wife, Kira, coined to describe the merging of TEXnology with Acrobat technology (Acronology?). The path from a `tex` file to a `pdf` file is a natural one:



TEX is ideally suited for creating `pdf` files, especially technical material. With the aid of the very powerful macro facility and the ability to position material very precisely on a page, hypertext links, and more importantly, form fields can be created and placed in an exacting and automated way.

[1] The other possibility is TEXrobat!

As I am primarily interested in promulgating TEX as a natural vehicle for producing `pdf` files, the examples in this article are presented using TEX code. For example, the following code defines a hypertext jump to the named destination, `TargetPage`, located within this document. All examples in the electronic version of these notes work, so just click on "Click Here."

Click Here

```
\leavevmode\htxt{\hg{Click Here}}\special{ps: %
[ /Rect \Rect                % /Rect [ xll yll xur yur ]
  /Border [ 0 0 0 ]
  /Action /GoTo
  /Dest /TargetPage
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

The TEX macros `\htxt`, `\hg` and `\Rect` are defined below.

- **PostScript Users**

PostScript users need not fear, however. Included within each example is the pure PostScript code for making the annotation. Consider the example below; I have put the `PostScript` portion in blue. A person wanting to use this code in a non-TEX document need only highlight the blue portion with the Acrobat Reader's **Select Text** under the

Tools menu, copy it to the clipboard, and paste it into the target document.

```
\htxt{\hg{Click Here}}\special{ps: %
[ /Rect \Rect                          % /Rect [ xll yll xur yur ]
  /Border [ 0 0 0 ]
  /Action /GoTo
  /Dest /TargetPage
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

However, I will not bother to highlight the `PostScript` portion of the code again. Just copy everything from the opening left bracket `[` to the closing `/ANN pdfmark`.

One note, it is necessary to calculate the dimensions of the bounding rectangle. In TEX, I use the macro `\Rect` to do this. The macro `\Rect` ultimately expands to the proper syntax needed by the `/Rect` key:

`/Rect \Rect` expands to `/Rect [ xll yll xur yur ]`

where `xll` and `yll` are the xy-coordinates of the lower-left corner of the bounding rectangle, and `xur` and `yur` are the xy-coordinates of the upper-right corner of the bounding rectangle. These numbers must be supplied by the document author, that's you.

After you've copied and pasted the above example, it might look like this in your PostScript source file:

```
[ /Rect [ 277 708 335 716 ]
  /Border [ 0 0 0 ]
  /Action /GoTo
  /Dest /TargetPage
  /Subtype /Link
/ANN pdfmark
```

- **TEX Users**

These notes are written using $\mathcal{AMS}$-TEX; anything having to do with `pdfmarks`, however, uses only TEX primitives; hence, can be used with any macro package.

The content of these articles will be most beneficial to individuals who want to create their own technical materials, write their own macros to have total and complete control over their document format, and explore and extend the capabilities of AcroTEX all within the context of WEB and/or CD-ROM publishing.

There will be many LaTEX2e users out there who may wonder, "What good are these notes? I can simply use Sebastian Rahtz' `hyperref` package." And indeed you can . . . for the links portion of these notes.

(If you use `hyperref`, at the minimum, these notes may represent a more pleasant alternative to reading [pdfm].) For forms, I don't believe Sebastian has written a '`hyperforms`' package quite yet. (Could it be his next project?)

I started writing hypertext electronic documents *before* the creation of `hyperref` anyway. At the time, my computer was rather slow and low on memory; it was my decision *not* to use LaTeX2e for that reason. I wanted to TeX as quickly as possible to correct the '$\infty - \epsilon$' number of errors.

Neophyte that I was, I was unaware of Acrobat (Version 1.0) at the time I started to write e-Calculus and had originally planned to create a `dvi` file based tutorial on our department's Intranet. This was why I chose the Y&Y TeX System: They offered the only manifestation of TeX that had hypertext from the `dvi` previewer, DVIWindo.

When my eyes finally turned to the Internet and Acrobat, I found myself in a very good position using the Y&Y System. Their use of Type 1 Fonts as their 'native font' and the automatic conversion from their hypertext links to `pdfmarks` put me miles ahead of the competition. (Note: There is no competition, but it put me miles ahead anyway.)

- **Configuring the Dvi to PostScript Driver**

In order to make the examples listed below work properly, you must configure your dvi-to-PostScript driver, DVIPSONE or DVIPS, properly.

DVIPSONE. This is the driver that I use with my Y&Y System. I use the '`\special{ps: <ps text>}`' throughout the examples, this requires the '`-j`' switch on DVIPSONE:

```
dvipsone -j <filename.dvi>
```

The command-line setting for DVIPSONE that I have been using is

```
dvipsone -v -k -*c -j -d=\distasst.ps <filename.dvi>
```

where

`-v` is for verbose mode.

`-k` Assume all fonts are printer resident. This means that the fonts are not written to the `PostScript` file making a smaller `PostScript` file. Distiller must now find the fonts in `c:\psfonts`.

`-*c` Allows the use of the `colorimage` operator for `TIFF` files;

`-j` Allows verbatim use of `PostScript` code.

`-d=\distasst.ps` This directs the output to the file `\distasst.ps` in the root directory where Distiller Assistant can detect its presence and activate the Distiller.

Dvips. The freeware TEX System MikTeX 1.10 [miktex] was downloaded from **CTAN** and installed on my computer (Win95). MikTeX uses DVIPS 5.76 from Radical Eye Software [dvips]. *It is important to use* **Type 1 fonts** *if you want to create a quality* `pdf` *document.* Fortunately, MikTeX comes with **Type 1** fonts supplied by Bluesky/Y&Y/AMS et al which I found very easy to set up just by editing the `config.ps` file. (The instructions are contained in the `config.ps` file itself.)

The only switch needed was `-M`. This commands `dvips` not to include bitmapped fonts in the `PostScript` file:

```
dvips -M <filename.dvi>
```

Since I use the Acrobat Distiller, the command line that I actually use is

```
dvips -M -o c:\distasst.ps <filename.dvi>
```

This names the output `PostScript` file as `distasst.ps` and places it in the root directory of my `c:` drive. The Distiller Assistant then detects its presence and calls the Distiller.

1.2. Some TEX Macros for Links

Let us begin by introducing some basic registers and macros needed to construct the links and form fields detailed below.

```
%
% Switch to distinguish between dvipsone and dvips
\newif\ifdvipsone \dvipsonetrue   % I use DVIPSONE
%
% \bbox holds the text that is to be defined as a hypertext link.
% The box has height (\ht\bbox), width (\ht\bbox), depth (\dp\bbox).
\newbox\bbox \newdimen\tmpdimen  % scratch dimension register
%
% Registers to transfer dimensions of \bbox to count registers.
\newcount\hhght \newcount\hwdth \newcount\hdpth
%
% Put the hypertext in \htxt, then transfer dimensions to \hhght,
% \hwdth, and \hdpth.
\def\htxt#1{\setbox\bbox=\hbox{#1}\hhght=\ht\bbox\hwdth=\wd\bbox
   \hdpth=\dp\bbox}
%
% PostScript Prologue. The dvips version was written by
% David Wilson of Monash University, Australia
\ifdvipsone
\special{!/TeXtoPDF {65536 div mag 1000 div mul} def  % sp to pts
          /PDFtoTeX {65536 mul mag 1000 div div} def} % pts to sp
\else % dvips
```

```
\special{!userdict begin
   /TeXtoPDF {65536 div DVImag mul} def          % sp to pts
   /PDFtoDvips {72.27 div Resolution mul} def    % pts to pixels
   /PDFtoVDvips {72.27 div VResolution mul} def  % pts to pixels
   /DvipstoPDF {72.27 mul Resolution div} def    % pixels to pts
   /HTeXtoDvips {TeXtoPDF PDFtoDvips} def        % sp to pixels
   /VTeXtoDvips {TeXtoPDF PDFtoVDvips} def end}  % sp to pixels
\fi
%
% Compute dimensions of bounding rectangle: /Rect [ llx lly urx ury ]
% The dvips version: by David Wilson, Monash University, Australia.
\ifdvipsone
% DVIPSONE expects coordinates of /Rect to be in \TeX's scaled points
\def\Rect{%
  [ currentpoint 2 copy \the\hdpth\space add 4 2 roll      % y1 and x1
    exch \the\hwdth\space add exch \the\hhght\space sub ]} % x2 and y2
\else
% dvips expects coordinates of /Rect to be in "pixels" (dots?)
\def\Rect{%
  [ currentpoint 2 copy \the\hdpth\space VTeXtoDvips add   % y1 and x1
    4 2 roll exch \the\hwdth\space HTeXtoDvips add         % x2
    exch \the\hhght \space VTeXtoDvips sub ]}              % y2
\fi
```

That ends the basic set of macros used for creating annotations in the `pdf` file. Hopefully, simplicity itself![2]

Now let's have a little Color.

```
%
% Some macros for producing color text
%
\def\pushcolor#1{\special{color push rgb #1}}
\def\popcolor{\special{color pop}}
\def\green{0 0.5 0} \def\blue{0 0 1}      % a limited number of
\def\red{1 0 0} \def\brown{.6 0 0}        %   basic colors
\def\hb#1{\pushcolor\blue{#1}\popcolor}   % blue text
\def\hg#1{\pushcolor\green{#1}\popcolor}  % green text
\def\hr#1{\pushcolor\red{#1}\popcolor}    % red text
\def\hbr#1{\pushcolor\brown{#1}\popcolor} % brown text
```

The specials

```
\special{color push rgb < r g b >}
```

and

```
\special{color pop < r g b > }
```

[2]Note that the macros for DVIPSONE are much simpler than those for dvips. This is because the Y&Y System and DVIPSONE were designed around the Type 1 font, while dvips was originally based on the pk font.

are available with DVIPSONE, the print driver supplied by Y&Y, as well as with DVIPS. You can implement these little macros, or use some packaged macros available with LaTeX2e.

These are pretty much all the macros you need[3] to create any type of link or form field you may want. All the above macros (with the exception of the macros specialized to DVIPS) are the ones I use on my Y&Y TeX System and have been proven to be very reliable.

1.3. Actions

Associated with any link is some sort of action. The [pdfs, p. 100] lists the possible types of actions: `GoTo`, `GoToR`, `URI`, `Launch`, `Thread`, `Sound`, `Movie`, `SetState`, `Hide`, `Named` (actions), `SubmitForm`, `ResetForm`, and `ImportData`.

In this article, we concentrate (almost) exclusively on `GoTo`, `GoToR`, and `URI` as actions with a few examples of `Named` actions. In the article on Forms, we shall have examples of `Hide`—a very useful action—and `ResetForm`.

[3] Additional specialized macros are introduced the article, however.

In Thomas Merz's Pdfmark Primer, [primer], most all the above actions are discussed in some detail.

- **Two Methods for Creating Action Links**

Throughout the links portion of Links and Forms, I will be presenting two methods of associating actions with hypertext links. The first method is the one described in [pdfm], and is the one usually implemented; the second method is referred to as "custom actions." In my experimentations, I have discovered that this syntax is not reserved only for special "custom actions," but can be used, in general, for any action.

Standard Method

Click Here

```
\htxt{\hg{Click Here}}%
\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action /GoTo
  /Dest /TargetPage
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Custom Action

Click Here

```
\htxt{\hg{Click Here}}%
\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action << /Subtype /GoTo
             /D (TargetPage) >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

In the second method, the value of the `/Action` key is a *dictionary* of information. **Note:** Instead of writing `/Subtype /GoTo`, one can also write `/S /GoTo`, which is the "filtered form."

There are several reasons one might prefer this custom action method:

1. It can be used for *all actions*.
2. Some actions—such as Named Actions and URL's—*require* this syntax.
3. The custom action method is needed if it is desired to create multiple actions.
4. This syntax is closer to what eventually appears in the `pdf` file. **Note:** The action key `/Action` gets filtered into `/A`, but it is incorrect to use the `/A` key within your source code.
5. This is the syntax *required* for creating form fields (such as buttons) with actions attached. **Note:** The `/Action` key is *incorrect* in this case; we *must use* the filtered form, `/A`.

1.4. Link Appearance

Before we get into surveying the various annotations, let's take a moment out to survey the many ways of highlighting the hypertext button. These methods are valid for all annotation of `/Subtype /Link`.

The following example is the simplest type of link, the Go To View. We'll use it to illustrate how to give a link different "appearances."

Example 1.1. Default Appearance. Jump to page 3 of the current document. The default appearance of the link is a black bounding rectangle.

▶ Jump to page 3

```
\htxt{Jump to page 3}\special{ps: %
[ /Rect \Rect
  /Page 3              % jump to page 3
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example 1.1. ■

Example 1.2. Invisible Rectangle. Jump to page 3 of the current document. The default appearance of the link is a black bounding rectangle.

▶ Click Here!

```
\htxt{Click Here!}\special{ps: %
[ /Rect \Rect
  /Page 3              % jump to page 3
  /Border [ 0 0 0 ]  % turn border off (white border)
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: If you turn the bounding rectangle off, then it is necessary to highlight the hypertext button using some other mechanism such as colored text. Example 1.2. ■

Example 1.3. Changing the Color of Border. The color of the box is controlled by the key-value pair `/Color [ r g b ]`, where $0 \leq$ `r` ≤ 1, $0 \leq$ `g` ≤ 1, and $0 \leq$ `b` ≤ 1. Here is a jump highlighted by a red bounding rectangle.

▶ Jump to page 3 one more time!

```
\htxt{Jump to page 3}\special{ps: %
[ /Rect \Rect
  /Page 3
  /Color [ 1 0 0 ]        % red bounding box
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox{} one more time!
```

We can use our TEX macros to enter the color for us. Recall that we have defined `\green` to be `\def\green{0 0.5 0}`. Thus,

▶ Jump to page 3. Ho, hum!

```
\htxt{Jump to page 3.}\special{ps: %
[ /Rect \Rect
  /Page 3
  /Color [ \green ]        % green bounding box
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox{} Ho, hum!
```

Example 1.3. ■

Example 1.4. Thicker Borders. The default width of the rectangular border is 1pt. This can be changed using the third parameter of the `/Border` key word. The syntax is `/Border [ 0 0 w ]`, where `w` is the width of the border in printer's points. `/Border [ 0 0 1 ]` is the default setting. Consider the following examples:

▶ Jump to page 3 Check out that Introduction!

```
\htxt{Jump to page 3}\special{ps: %
[ /Rect \Rect
  /Page 3
  /Color [ 1 0 0 ]              % red bounding box
  /Border [ 0 0  2 PDFtoTeX ]  % /Border [ 0 0 2 ]                    (*)
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox{} Check out that \hb{Introduction}!
```

Example Notes: DVIPSONE: The PostScript procedure `PDFtoTeX` in line `(*)` converts its argument (2 in this case) to scaled points which DVIPSONE operates in. Now DVIPSONE can, in turn, convert this value back to printers points (which is 2). (Pooh.)

- Dvips: In place of line `(*)`, use `/Border [0 0 2 PDFtoDvips]`

This converts points to "pixels" that DVIPS expects such numbers to be measured in ... now DVIPS converts this back to points! (Double Pooh!) 𝔇𝔓 ■

The example with the thicker border does not have a very good look to it. Let's make some adjustments. One possibility is to insert a `\strut` and a couple of `\thinspaces`.

▶ Jump to page 3

```
\htxt{\,\strut Jump to page 3\,}\special{ps: %
[ /Rect \Rect
  /Page 3
  /Color [ 1 0 0 ]              % red bounding box
  /Border [ 0 0 2 PDFtoTeX ]    % /Border [ 0 0 2 ]
  /Subtype /Link                % Note: dvips use `2 PDFtoDvips'
/ANN pdfmark}\unhbox\bbox
```

Example Notes: The macro '`\,`' in $\mathcal{AMS}$-TEX is a thinspace. In TEX, a thinspace is '`\thinspace`', but this thinspace does not operate in math mode as '`\,`' does. ∎

Another way to create an 'eye-pleasing' thick rectangle—or for any rectangle for that matter—is to design an adjustment mechanism built into the `\htxt` macro.

I don't need this particular feature, but it seems to be a good idea. Perhaps you will find it useful.

```
\begingroup   % begin special temporary definition for \htxt
%
%  Usage:
%  \htxt[#1,#2,#3]#4 : typeset #4
%  increase \ht\bbox by #1, increase \wd\bbox by twice #2,
%  and increase \dp\bbox by #3.
\newdimen\tmpdimen  % temporary scratch dimension register
%
% Define a new \htxt with thee new arguments
\def\htxt{\futurelet\next\htxti}
\def\htxti{\ifx\next[\expandafter\htxtii\else\expandafter\htxtiii\fi}
\def\htxtii[#1,#2,#3]#4{\setbox\bbox=\hbox{\kern#2#4\kern#2}%
   \adjustbbox[#1,#3]\hhght=\ht\bbox\hwdth=\wd\bbox\hdpth=\dp\bbox}
\def\htxtiii#1{\setbox\bbox=\hbox{#1}\hhght=\ht\bbox
   \hwdth=\wd\bbox\hdpth=\dp\bbox}
```

```
\def\adjustbbox[#1,#2]{\tmpdimen=\ht\bbox\advance\tmpdimen#1
   \ht\bbox=\tmpdimen\tmpdimen=\dp\bbox\advance\tmpdimen#2
   \dp\bbox=\tmpdimen}
```

Now let's use the new `\htxt` macro to increase the dimensions of the bounding rectangle by 3pts in each direction.

▶ Jump to page 3

```
\htxt[3pt,3pt,3pt]{Jump to page 3}\special{ps: %
[ /Rect \Rect
  /Page 3
  /Color [ 1 0 0 ]              % red bounding box
  /Border [ 0 0 2 PDFtoTeX ]    % /Border [ 0 0 2 ]
  /Subtype /Link                % Note: dvips us `2 PDFtoDvips'
/ANN pdfmark}\unhbox\bbox
\endgroup    % special definition for \htxt
```

Example 1.4. ■

Example 1.5. Dashed Borders. The solid rectangular border can be changed to a dashed border using the `/Border` key word.

▶ Jump to page 3

```
\htxt{\,\strut Jump to page 3\,}\special{ps: %
[ /Rect \Rect
  /Page 3                              % jump to page 3
  /Color [ 1 0 0 ]                     % red bounding box
  /Border [ 0 0 1 PDFtoTeX [3] ]       % dashed border, 1 pt wide
  /Subtype /Link                       % Note: dvips use '1 PDFtoDvips'
/ANN pdfmark}\unhbox\bbox
```

Example Notes: The syntax for a dashed border is

`/Border [0 0 w [d] ]`

where `w` is the width of the border measured in points, and `[d]` is a dash array that specifies the length of the dashes and gaps in the link's border. The maximum value of `d` is 10. [pdfm, p. 10].

■ Note that I have increased the size of the bounding rectangle by including a `\strut` and two `smallspaces` (`\,`). ■

Let's finish up this example by looking at a dashed bounding rectangle two points in thickness (medium thickness in Exchange), with the `d` = 10 (dashes and gaps at a maximum).

▶ Jump to page 3

```
\htxt{\,\strut Jump to page 3\,}\special{ps: %
[ /Rect \Rect
  /Page 3                              % jump to page 3
  /Color [ 1 0 0 ]                     % red bounding box
  /Border [ 0 0  2 PDFtoTeX [10] ]     % dashed border, 2 pt wide
  /Subtype /Link                       % Note: dvips use ‘2 PDFtoDvips’
/ANN pdfmark}\unhbox\bbox
```

Example 1.5. ■

Example 1.6. Highlighting the Button. When a hypertext button is pressed, the Acrobat Reader can highlight the button in four ways: None, Invert (the default), Outline, and Inset.

▶ No Highlighting

```
\htxt{\,\strut No Highlighting\,}\special{ps: %
/ANN pdfmark}\unhbox\bbox
  /Page 1                  % jump to page 1
  /Color [ 0 1 0 ]         % green bounding box
  /H /N                    % highlighting set to none.
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: I find this variation useful when using a `tiff` picture, giving an icon effect, as a hypertext button. ■

The next variation is invert highlighting. This is the default setting, hence, the inclusion of the key-value pair `/H /I` is not necessary.

▶ Invert Highlighting

```
\htxt{\,\strut Invert Highlighting\,}\special{ps: %
/ANN pdfmark}\unhbox\bbox
  /Page 1                 % jump to page 1
  /Color [ 0 0 1 ]        % blue bounding box
  /H /I                   % highlighting set to Invert.
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Outline highlighting may have use when framing an icon button, but not in this example!

▶ Outline Highlighting

```
\htxt{\,\strut Invert Highlighting\,}\special{ps: %
/ANN pdfmark}\unhbox\bbox
  /Page 1                 % jump to page 1
  /Color [ 0 0 1 ]        % blue bounding box
  /H /O                   % highlighting set to Outline.
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Outlining seems to have greater visual impact when the bounding rectangle is dashed.

▶ Outline Highlighting

```
\htxt{\,\strut Invert Highlighting\,}\special{ps: %
/ANN pdfmark}\unhbox\bbox
  /Page 1                      % jump to page 1
  /Color [ 0 0 1 ]             % blue bounding box
  /Border [ 0 0 65536 [3] ]    % dashed border
  /H /O                        % highlighting set to Outline.
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Finally, there is inset highlighting. I'm not quite sure of what to make of this method of highlighting.

▶ Inset Highlighting

```
\htxt{\,\strut Invert Highlighting\,}\special{ps: %
/ANN pdfmark}\unhbox\bbox
  /Page 1                      % jump to page 1
  /Color [ 1 0 0 ]             % red bounding box
  /H /P                        % highlighting set to Inset.
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example 1.6. ■

Example 1.7. Coloring the Text. A very common method of creating a link appearance is by coloring the target text. In this case, an invisible rectangle is used.

▶ Click here.

```
\htxt{\hg{Click here.}}\special{ps: %
/ANN pdfmark}\unhbox\bbox
  /Page 1                     % jump to page 1
  /Border [ 0 0 0 ]           % invisible border
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

In my tutorials, I use two colors for hypertext links: green and brown. Green is used to denote jumps to within the document, and brown denotes jumps to another document. When browsing `pdf` files on the Web, there is a significant difference in download time between these two link types. The user may not be so eager to follow a brown link.

▶ Click here

```
\htxt{\hbr{Click here}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]           % turn border off
  /Action /GoToR              % Go to Remote
  /Page 1                     % Go to first page of...
  /File (examples/sample.pdf) %   sample file
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example 1.7. ■

Throughout the rest of this article, I will use the green/brown linking convention.

1.5. Within Document Jumps

For annotations of `/Subtype /Link`, the actions to be performed are determined by the value of the `/Action` key. In these notes, we will look at two actions in particular: `/Action /GoTo` and `/Acton /GoToR`. For a complete listing of all permissible actions, see [pdfs, p. 96] or the [primer].

Example 1.8. Jump to a Page Number. Jump to page 3 by clicking anywhere on the highlighted phrase "Jump to page 3" above.

```
\htxt{\hg{Jump to page 3}}\special{ps: %
[ /Rect \Rect
  /Page 3                % In general: /Page N
  /Border [ 0 0 0 ]      % invisible rectangle
  /Action /GoTo          % go to view
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Actually, the `/Action /GoTo` key-value pair is the default action and is therefore not needed in the above code. Consider: Jump to page 3 again.

```
\htxt{Jump to page 3}\special{ps: %
[ /Rect \Rect
  /Page 3                % In general /Page N
  /Border [ 0 0 0 ]      % invisible rectangle
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: More generally, to jump to page `N`, replace `/Page 3` with `/Page N`. ■

There is an alternate syntax that works as well. This syntax comes in handy when you want to perform multiple actions (Section 1.10):

▶ Jump to page 3 (1)

```
\htxt{Jump to page 3}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /Subtype /GoTo /D [ {Page3} /XYZ null null null ] >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: The key-value pair

```
/D [ {Page3} /XYZ null null null ]
```

defines the destination of the jump: page 3 with zoom factor inherited from the current page. Techniques for controlling the view are discussed in Section 1.8, page 53.

▪ To jump to page 10, use `{Page10}` instead; in general, to jump to page `N`, use `{PageN}`.

▪ Using this style—so-call customized actions—the keys must be in their "filtered" forms, `/D` instead of `/Dest`, for example, We can also filter `/Subtype /GoTo` key-value to `/S /GoTo`. (The Distiller will accept either `/Subtype` or `/S` here, but will not take `/Dest` in place of `/D`.) Example 1.8. ■

Example 1.9. Jump to Relative Page. According to [pdfm, p. 29] we can jump to previous or next page.

▶ Jump to Next Page

```
\htxt{Jump to Next Page}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Page /Next           % For previous page, use /Page /Prev
  /Action /GoTo
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: Possible values of `/Page` are `/Next` and `/Prev`. If the destination is the current page, then the `/Page` key is *omitted*. Naturally, the `/Page` key takes a number for a value, as discussed in Example 1.8. ■

The custom action version of these same link is

▶ Jump to Next Page

```
\htxt{Jump to Next Page}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /GoTo /D [ {NextPage} /XYZ null null null ] >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: Possible values for page objects are `{NextPage}`, `{PrevPage}`, `{ThisPage}` and, of course, `{PageN}`, as discussed in Example 1.8.

- The page object `{ThisPage}` is useful for changing the view of the current page. See the Section 1.8 on controlling the view.
- The expression `/XYZ null null null` controls the view of the page being jumped to. In this case, the current view is preserved. See Section 1.8, page 53, for more details on controlling the view. ■

Alternatively, by [pdfs, p. 110], we can use Named Actions to jump to the first page, last page, next page and previous page.

Named actions are typically used to execute menu functions; additional examples of named actions can be found in Section 1.9, page 70.

▶ Go to the last page (2)

```
\htxt{\,\strut Go to the last page\,}\special{ps: %
[ /Rect \Rect
  /Color [1 0 0]                          % red box
  /Action << /S /Named /N /LastPage >> % last page
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: The documented values of the `/N` key are

`/N /FirstPage` to go to the first page;
`/N /LastPage` to go to the last page;
`/N /NextPage` to go to the next page;
`/N /PrevPage` to go to the previous page.

▪ Most all of the menu functions of Reader and Exchange can be executed using named actions. The values of the `/N` key for the various menu functions are mostly (officially) undocumented; fortunately, Thomas Merz has cataloged virtually all of them! For a complete listing, see the Pdfmark Primer, Chapter 6, Table 6.31.

▪ There will be a brief discussion of named actions as well as a few examples in Section 1.9, page 70. Example 1.9. ■

Before we can begin our survey of jumping to named destinations, we first must learn how to "plant" one of these markers. The following illustrates how you can define a named destination

Example 1.10. Defining a Named Destination. For people writing their source file in TEX or PostScript, it pretty easy to define this type of `pdfmark`. The reader is referred to the Pdfmark Primer for discussions on how to define destinations when authoring with Adobe FrameMaker or Microsoft Word. On the other hand, such applications may automate the process of defining destinations for such document elements as chapter headings and sections.

▶ Simple Destination.

```
\special{ps: %
[ /Dest /MyFavoriteDest
  /DEST pdfmark}%
```

This establishes a destination that we can jump to—and we will in the next example.

▶ Destination with a View.

```
\special{ps: %
[ /Dest /MyFavoriteView
  /View [ /XYZ null null 4 ]
  /DEST pdfmark}%
```

Example Notes: This destination, or anchor, has a magnification factor associated with it. Should we jump to this reference destination, we would arrive with a zoom factor of 4 (400%).

■ See Section 1.8 for more information on the meaning of the `/View` key. ■

You can also define destinations using a `/Page` key; in this case, a destination created on one page, defines a destination on another.

▶ Destination with `/Page` key. (3)

```
\special{ps: %
[ /Dest /HomeSweetPage
  /Page 1
  /DEST pdfmark}%
```

Example Notes: This effectively defines an alias between the first page of the document and the value `/HomeSweetPage`. This mark will be illustrated in the next example as well.

■ Of course, jumping to page 1 can be accomplished by using the methods of Example 1.8, so I remain unconvinced of the need for this particular `/Page` feature. I really haven't been able come up with a good example of the need to define destinations in this way. ■

The `/DEST pdfmark` can be packaged into a TEX macro. Here's one suggestion:

```
%
\def\hmark#1{\leavevmode\special{ps: [ /Dest /#1 /DEST pdfmark}}
%
```

Y&Y has a built-in `\special` for creating destinations, or 'marks' as they call them.

```
%
% Y&Y only: \special{mark: <dest> }
%
\def\hmark#1{\leavevmode\special{mark: \space #1}}
%
```

Macro Notes: This last macro is the one I use in my own TEX source files.

■ See SECTION 1.13 for a detailed discussion of the hypertext capabilities of the TEX system developed by Y&Y. Example 1.10. ■

Let's turn now to, admittedly, the most important type of jump: jumping to a *named destination*. This type of jump is not supported by Exchange in the sense that you cannot use Exchange to define a jump to a named destination; nonetheless, for authors who have control over the source file, this method stands as a power method of defining a hypertext jump.

Example 1.11. Jump to a Named Destination. In this example, we illustrate the code required to jump to a named destination. As was mentioned before, PostScript users can simply copy and paste the relevant parts into their document, and modify as desired.

For authors using high-power applications, special techniques are generally required. The interested reader should refer to Section 6.3 of the Pdfmark Primer for more information on this topic.

▶ My Favorite Destination

```
\htxt{\hg{My Favorite Destination}}\special{ps:
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Dest /MyFavoriteDest
  /Action /GoTo
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

This jumps to the destination without changing the magnification of the document. Contrast this last jump with the following one:

▶ My Favorite View

```
\htxt{\hg{My Favorite View}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Dest /MyFavoriteView        % same page, but with mag 4
  /Action /GoTo
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

To remind you again, the `/GoTo` action is the default, hence, not needed when you jump to a destination *within the same document*.

▶ My Favorite Destination

```
\htxt{\hg{My Favorite Destination}}\special{ps:
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Dest /MyFavoriteDest
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

You can also use the syntax of "customized actions" to execute the jump. The is useful when it is necessary to perform multiple actions.

▶ Go to Target Page

```
\htxt{\hg{Go to Target Page}}\special{ps:
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /GoTo /D (TargetPage) >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: Here, the destination key `/Dest` gets filtered to simply `/D` and the *named target* is specified by a *string*: `(TargetPage)`.

- According to [pdfs, p. 100], "There are several performance advantages to using strings instead of names for named destinations."

It goes on to say, [pdfs, p. 277], the use of strings does not put a limitation on the number of destinations that can be defined. ■

I almost forgot. Let's check out our `/HomeSweetPage` destination mark that we defined in line (3) as a part of Example 1.10. This mark was defined on page 39, but we wanted it to reference page 1. Let's see if it worked.

▶ Go to Home Sweet Page

```
\htxt{\hg{Go to Home Sweet Page}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action << /S /GoTo /D (HomeSweetPage) >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Finally, the appearance of the link—bounding rectangle, dashed versus solid, highlighting and coloration—can be changed using the methods enumerated in Section 1.4. Example 1.11. ■

1.6. Jumping to a Remote File

Creating a hypertext link to another document is quite similar to the `/GoTo` view. Here, we need the `/GoToR` key word (go to remote), the `/File` key (file specification) and *either* a page number key `/Page` *or* a destination key `/Dest`.

Important. Generally, all the links described in this section work on a disk-based system (off-line) or on the Web—as long as the Reader is acting as a plug-in—provided the file specification is given using *relative paths*. ■

Example 1.12. Go to a File. The following code opens a `pdf` file to the first page.

▶ Click Here

```
\htxt{\hbr{Click Here}}\special{ps: % brown for remote link
[ /Rect \Rect
  /Border [ 0 0 0 ]                 % invisible rectangle
  /Action /GoToR                    % go to a remote file
  /Page 1                           % go to page 1
  /File (examples/sample.pdf)       % relative path to file
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: This code works on a disk-based system, and, if the Reader is acting as a plug-in to one of the major browsers, works on the Web as well—as long as the file specification uses *relative paths*. ■

Here is the custom action variation on the same example.

▶ Click Here

```
\htxt{\hbr{Click Here}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action << /S /GoToR /D [ 0 /Fit ] /F (examples/sample.pdf) >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: Notice that the `/File` key has been filtered to `/F`.

▪ When using the "customized action," the value of the destination key /D is an array the first element of which is the page number. In this form, Page 1 corresponds to 0. (Page numbers are 0 based.) More on this in the next section. Example 1.12. ■

Example 1.13. Go to a Page in a File. To jump to a selected page, just include the `/Page` key with value equal to the desired page example.

▶ Introduction

```
\htxt{\hbr{Introduction}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ] % invisible rectangle
  /Action /GoToR                % go to remote file
  /Page 2                       % page 2, the introduction
  /File (examples/sample.pdf)   % relative path to file
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: This type of link would be useful if you wanted to link your file with another static file (perhaps created by another). In this case, you could reference a particular page of interest without fear that the page number would change.

▪ This code will work properly off-line on a local disk system, or on-line through the Web. The file specification is expressed as a *relative path* to the file.

▪ I haven't found a need for this particular type of link; in my file system, the documents are always undergoing revisions. ▪

Here's a version of this same link written using "customized actions."

▶ Go to Page 2: Introduction

```
\htxt{\hbr{Go to Page 2: Introduction}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /GoToR /D [ 1 /XYZ null null null ] % 1 = page 2
     /F (examples/sample.pdf) >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: The destination key `/D` allows indirect references to Page Objects in the case of `/GoTo` (See (1), EXAMPLE 1.8); however, in the case of `/GoToR`, it is explicitly mentioned in [pdfs, p. 104] that

the destination page, the first element in the array, must be specified as a page number, *not* by an indirect reference to a Page Object.

- **Important! The first page is page 0.**
- Hence, if we want to jump to page 2, we specify a 1 for the page destination parameter. ■

One could obviously implement a simple macro to make the page calculation.

▶ Go to Page 2: Introduction

```
%
\def\Page#1{#1\space 1 sub}
%
\htxt{\hbr{Go to Page 2: Introduction}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /GoToR /D [ \Page 2 /XYZ null null null ]
     /F (examples/sample.pdf) >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example 1.13. ■

Example 1.14. Go to a Named Destination. In a tutorial system, such as the ones I have been developing, it is absolutely essential to develop cross-referencing. The `/GoToR` with a named destination is the work-horse of any such system.

▶ Go to Target in Sample File

```
\htxt{\hbr{Go to Target in Sample File}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action /GoToR                % go to remote file
  /Dest /Target                 % named destination in ...
  /File (examples/sample.pdf)   %   examples/sample.pdf
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

Example Notes: Beginning with Acrobat Reader 3.0, this link works properly for both disk-based systems and Web systems. (As long as paths are *relative* and Reader is acting as a plug-in for the browser.)

▪ Prior to version 3.0, this link did not work on the Web. The browser would not go to the destination page, instead would open the default page, page 1. ▪

The customized action version of this link is . . .

▶ Go to Target in Sample File

```
\htxt{\hbr{Go to Target in Sample File}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /GoToR /D (Target) /F (examples/sample.pdf) >>
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

Example Notes: Notice once again the key names have been filtered: `/S` instead of `/Subtype`, `/D` instead of `/Dest`, and `/F` instead of `/File`.

■ Important! Note also that the destination is given as a (PostScript) string rather than a name, (I.e. `(Target)` instead of `/Target`.) Whereas, Distiller will not choke on `/D /Target`, the resulting link does not jump the correct page in the sample file—it goes to the default page, page 1. ■

Recall: The appearance of the link—bounding rectangle, dashed versus solid, highlighting and coloration—can be changed using the methods enumerated in Section 1.4.

Byte Serving. When jumping to a named destination in a remote file, the Web server which is byte-serving enabled, will return only the requested page—not the whole document. Example 1.14. ■

Example 1.15. Go to an URL. This kind of link is needed if you want to reference `pdf` files *or* `html` files on the Web. For an alternate approach using the `Launch` key, see Example 1.16.

Occasionally, you might want to reference a document on *another server*. In this case, the `/GoToR` key does not work. To reference a full URL, a new customized action scheme requiring a special `/Subtype /URI` is needed.

▶ Click Here

```
\rtr\htxt{\hbr{Click Here}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /URI   % = /Subtype /URI
  /URI (http://www.math.uakron.edu/\noexpand~dpstory/%                (*)
        tutorial/pdfmarks/examples/sample.pdf\#Target) >>            (**)
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

Example Notes: Important! Note the method of specifying a named destination in line `(**)`. (Quite similar to specifying anchors in `html` files). PostScript users should read "`.../sample.pdf#Target)`." The pound symbol '`#`' has a special meaning. To typeset a '`#`' it is necessary to type '`\#`'.

- PostScript users should ignore the `\noexpand` in line `(*)`. The tilde (`~`) is an "active" symbol in TEX. The `\noexpand` temporarily keeps TEX from expanding on the meaning of the `~`. In line `(*)`, just read `http://www.math.uakron.edu/~dpstory/`.
- By the way, the comment character '`%`' is needed at the end of line `(*)`; this keeps unwanted space characters from creeping into our Web address.

▪ Under the menu File > Preferences > Weblink..., adjust the list box to 'Always Show.' Having done this, the URL of any Web link will be shown in the help bar at the bottom of the Reader window.

▪ This URL *is not shown* when the Reader is acting as a plug-in to one of the Web browsers. ▪

You can also request `html` documents using this same customized action syntax.

▶ e-Calculus

```
\htxt{\hbr{\eCalculus}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /URI   % = /Subtype /URI
  /URI (http://www.math.uakron.edu/\noexpand~dpstory/%
      e-calculus.html) >>
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

Example Notes: The `\eCalculus` TEX macro is the one I use to type set e-Calculus in the Viva Regular Font, with kerning. ▪

Recall: The appearance of the link—bounding rectangle, dashed versus solid, highlighting and coloration—can be changed using the methods enumerated in Section 1.4.

Should you have any comments about these notes, please feel free to e-mail me at ...

▶ dpstory@uakron.edu

```
\htxt{\hbr{dpstory\char‘@uakron.edu}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action << /S /URI   % = /Subtype /URI
  /URI (mailto:dpstory\noexpand@uakron.edu) >>                    (*)
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

Example Notes: Again the '`/noexpand@`' is to avoid problems in TEX and to actually write '`@`' to the PostScript file. PostScript users, in line `(*)` read `/URI (mailto:dpstory@uakron.edu)`. Example 1.15. ■

1.7. The Launch Action

The Launch action can be used to go to an URL or to launch a program application on the local system. The latter is well known, but the former is not.

Example 1.16. Go to an URL. The following code opens a `html` in the default web browser. If the browser is not running, `Lanuch` will start it; if the link is clicked from within a browser, the browser will go to the link.

▶ Go Adobe Home Page

```
\htxt{\hbr{Go Adobe Home Page}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action /Launch
  /URI (http://www.adobe.com/)
  /Subtype /Link /ANN pdfmark}\unhbox\bbox
```

Example Notes: This `pdfmark` construct, when distilled, is the same as the distilled version of Example 1.15. Example 1.16. ■

1.8. Controlling the View

By including a view control in your hypertext jumps you can magnify a portion of the page that you want the reader's attention drawn to, or you can regain control of the document's fit in the window of the Reader.

When performing a hypertext jump to a page, fundamentally, there are two methods of controlling the view of that target page: (1) the control is put in placed on the jump side and (2) the control is placed on the destination side. The first method is available for backward

compatibility with Acrobat 1.0 and is somewhat limited in its applicability; the second is a much more powerful and more general method.

Example 1.17. /Fit. The `/Fit` key fits the page to the window. If the page is already "fit," this jump has no effect.

For within document jumps, the `/View` key can be used to define the view of the destination page.

In the code swatch that follows, click on the blue link—yes, I am breaking my green/brown convention here—first, this magnifies the page, then click on the green link.

▶ Click Here First ▶ Fit the Next Page!

```
\htxt{Fit the Next Page!}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Page /Next
  /View [ /Fit ]
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

▶ Click Here First ▶ Fit the Next Page!

```
\htxt{Fit the Next Page!}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /GoTo /D [ {NextPage} /Fit ] >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: This type of link is maintained for compatibility with Acrobat 1.0.

- Notice that the control is placed on the jump side of the link.
- For your reference, other values of the `/Page` key can be found in Example 1.8 and Example 1.9. ■

When jumping to a particular page within the current document, the above techniques work well, though knowledge of the page number *a priori* is rarely known. Jumping to named destinations is usually the preferred method; the `/View` key, however, *does not work* when there is a `/Dest` key. To verify the last statement, click on the following links. The second link jumps to `/Dest /Target` and includes a `/View [ /Fit ]` key value pair. ▶ Click Here First ▶ Will it Fit?

When jumping to a named destination, the control of the view shifts to the *destination page*. In this case, the `/View` key is used while defining

the destination mark.

Click on the links below, the destination page shows how to defined the mark that controls the view upon arrival.

▶ Click Here First Jump to Controlled View Page.

```
\htxt{\hg{Jump to Controlled View Page.}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action /GoTo
  /Dest /TargetPagewithFit
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: There is also the custom action version of the above link. To construct it, replace the `/Action` and `/Dest` key-value pairs with `/Action << /S /GoTo /D (TargetPagewithFit) >>`. ■

Now jump to another file.

▶ Click Here First ▶ Jump to Controlled View Page Elsewhere!

```
\htxt{\hg{Jump to Controlled View Page Elsewhere!}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action /GoToR
  /Dest /TargetwithFit
  /File (examples/sample.pdf)
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

The custom action version follows.

▶ Click Here First ▶ Jump to Controlled View Page Elsewhere!

```
\rtr\htxt{\hg{Jump to Controlled View Page Elsewhere!}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action << /S /GoToR /D (TargetwithFit) /F (examples/sample.pdf) >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example 1.17. ■

Example 1.18. /FitB. This key attempts to fit the bounding box of the page contents to the window. In effect, `/FitB` tends to reduce the amount of white space surrounding the text.

Experiment with the next link by reducing the size of the Reader (Browser) window and jumping to the next window. (E.g., fit the Reader window to fit the dimensions of this document.) Notice the page is magnified a little, and the edges of the page are out of view.

▶ FitB the Next Page!

```
\htxt{Fit\underbar B the Next Page!}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Page /Next
  /View [ /FitB ]
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: Other variations on the same theme can be obtained by repeating all the examples of EXAMPLE 1.17, with `/Fit` replaced by `/FitB`. Example 1.18. ■

Example 1.19. /FitH. The key-value pair `/FitH <top>` attempts to fit the *width* of the page to the window. The value `<top>` specifies the distance in default user space from the page origin to the top of the window.

Let's try this by displaying the bottom half of the next page. From Acrobat Exchange, we can read off some useful information needed to compute the lower half. From a standard 8.5" x 11", we have cropped up from the bottom by 396 pts. The cropped page itself is 353 pts high.

Inside the value of the `/View` key, we calculate the "distance from the page origin (lower left hand corner of the original page size) to the top of the window."

▶ FitH the Next Page! (4)

```
\htxt{Fit\underbar H the Next Page!}\special{ps: %
[ /Rect \Rect  /Border [ 0 0 0 ]          % combine two lines
  /Page /Next                            % go to next page
  /View [ /FitH 396 0.5 353 mul add ]    % <top> = 396 + 0.5 * 353
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

You may have to experiment with the window size to see this fit in action. Contrast this with the following example.

▶ FitH the Next Page!

```
\htxt{Fit\underbar H the Next Page!}\special{ps: %
[ /Rect \Rect  /Border [ 0 0 0 ]          % combine two lines
  /Page /Next                            % go to next page
  /View [ /FitH 792 ]                    % <top> = 792 pts = 11 inches
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: You get different results depending on the dimensions of you Reader/Browser window. The Reader will try to fit the specified height, (`<top>`) as close as possible, given the width of window.

■ See EXAMPLE 1.17 to see how to use this fit type with other types and styles of links. Just replace '`/Fit`' in these examples with '`/FitH <top>`' (with '`/FitH 0.5 353 mul add`', for example).

■ I haven't found much use for this fit type. Example 1.19. ■

Example 1.20. /FitBH. This fits the width of the bounding box of the page contents to the window. (`/FitBH` is a combination of `/FitB` and `/FitH`.) The syntax is `/FitBH <top>`, where `<top>` specifies the distance in the default user space from the page origin to the top of the window.

▶ FitBH the Next Page! (5)

```
\htxt{Fit\underbar{BH} the Next Page!}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Page /Next                                  % halfway up from bottom
  /View [ /FitBH 396 0.5 353 mul add ]         %   <top> = 396 + 0.5 * 353
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: Contrast this action with that of link at line (4) of Example 1.19. Notice especially the bottom edge of the window.

■ You get different results depending on the dimensions of you Reader/Browser window. The Reader will try to fit the specified height, (`<top>`) as close as possible, given the width of window.

- See Example 1.17 to see how to use this fit type with other types and styles of links. Just replace '`/Fit`' in these examples with '`/FitBH <top>`' (with '`/FitBH 0.5 353 mul add`', for example).
- I haven't found much use for this fit type. Example 1.20. ■

The vertical fit types `/FitV` and `/FitBV` are quite similar to their horizontal counterparts `/FitH` and `/FitBH`.

Example 1.21. /FitV. The key-value pair `/FitV <left>` attempts to fit the *height* of the page to the window. The value `<left>` is the distance in default user space from the page origin to the left edge of the window.

The page of this document has been cropped to be 396pts wide, we have cut 108pts from the *left* edge. Let's set `<left>` at a setting halfway across the page and see what we get.

You need to adjust your window to a narrow setting to see the effects of this jump.

▶ FitV to 2 Col Page, Left Col! (6)

```
\htxt{Fit\underbar V to 2 Col Page, Left Col!}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Dest /TwoColPageL          % destination mark has in it
  /Subtype /Link              %   /View [ /FitV 108 ]
/ANN pdfmark}\unhbox\bbox
```

▶ FitV to 2 Col Page, Right Col! (7)

```
\htxt{Fit\underbar V to 2 Col Page, Right Col!}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Dest /TwoColPageL          % destination mark has in it
  /Subtype /Link              %    /View [ /FitV 108 396 add ]
/ANN pdfmark}\unhbox\bbox
```

Example Notes: As these two examples suggest, a possible use of this fit type is to focus in on the left-hand side or the right-hand side of the page—possibly within the context of a two column format.

- I'm not sure of the true value of `/FitV`.

Example 1.21. ■

Example 1.22. /FitBV. The key-value pair `/FitBV <left>` attempts to fit the *height* of the bounding box of the page contents. The value `<left>` is from the page origin to the left edge of the window.

▶ FitBV the Next Page!

```
\htxt{Fit\underbar{BV} the Next Page!}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action << /S /GoTo /D [ {NextPage} /FitBV 180 ] >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

▶ FitBV the Next Page!

```
\htxt{Fit\underbar{BV} the Next Page!}\special{ps: %
[ /Rect \Rect /Border  [ 0 0 0 ]
  /Action << /S /GoTo /D [ {NextPage} /FitBV 180 396 2 div add ] >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: I can't really tell the difference between these two links. Neither of them "fits the height of the page to the window."

■ In this example, I've use the custom action syntax . . . just for a little change. Alternate syntax given in Example 1.21.

Example 1.22. ■

Example 1.23. /XYZ. The particular key `/XYZ` takes three arguments: `/XYZ <left> <top> <zoom>`. The values `<left>` and `<top>` specify the distance in default user space from the page origin to the top-left corner of the window. The parameter `<zoom>` is the desired *zoom factor* (Zoom factor of 1 corresponds to 100% magnification.)

Another allowable value of the parameters is `null`. A value of `null` for one of the parameters means that value will remain unchanged from the current setting.

The next example jumps to the fourth quadrant of the next page with 200% magnification.

▶ XYZ to Next Page!

```
\htxt{XYZ to Next Page!}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Page /Next
  /View [ /XYZ 108 396 add 396 353 2 div add 2 ]
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: Recall, 108pts have been cropped off the left vertical page and the cropped page itself is 396pts wide. I've also cropped 396pts off the bottom, and the cropped page is 353pts high. ■

According to [pdfm, p. 31], a common destination is the upper left hand corner with zoom factor 1. The manual goes on to recommend a 4pt separation between the window and the document. Let's have a manifestation of this statement.

▶ XYZ to Next Page!

```
\htxt{XYZ to Next Page!}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Page /Next
  /View [ /XYZ -4 4 11 72 mul add 1]
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: That doesn't look good. This jump would look much better if the document page was 8.5" x 11" with no cropping. ■

▶ XYZ to Next Page!

```
\htxt{XYZ to Next Page!}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Page /Next
  /View [ /XYZ null null null ]  % changing nothing!
  /Subtype /Link /ANN pdfmark}\unhbox\bbox
```

Example 1.23. ■

Example 1.24. /FitR. This fit style takes four arguments:

$$/\text{FitR } x_1 \ y_1 \ x_2 \ y_2.$$

`/FitR` fits the *rectangle* having lower left corner of x_1, y_1 and upper right corner of x_2, y_2 to the window.

In my opinion, `/FitR` and `/Fit` are the most useful of all the fit styles.

I've used `/FitR` several times in this article already. In the links Click Here First, such as the one in EXAMPLE 1.17, I used `FitR` to magnify the target word. Here's how I did it.

▶ Let me direct your attention to this Important Point.

```
\htxt{Important Point.}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Dest /GoToImportantPoint
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

"In conclusion, ladies and gentlemen, I have discovered the most marvelous and profound fact that I would like to share with this august and distinguished room of scholars. My little discovery is $1+1=2$, *a fact that cannot be refuted by arguments of reason*."

The equation $1+1=2$ has a destination mark just before it. Here is the code for that:

```
``...My little discovery is \htxt{\hb{$1+1=2$,}}\special{ps: %
[ /Dest /GoToImportantPoint
  /View [ /FitR \FitRbboxB5 ]                                    (*)
  /DEST pdfmark}\unhbox\bbox{} {\it a fact that cannot be refuted by
arguments of reason}.''
```

The destination mark has the `/View` key implanted with the `/FitR` fit style specification. The TEX macro, `\FitRbboxB` in `(*)`, calculates the coordinates of the bounding rectangle that contains the target word `\hb{$1+1=2$}`. This macro takes one argument: the 5 means the macro should include a 5pt boundary around the target word.

```
%
% \FitRbboxB : Calculates the x1, y1, x2, y2 coordinates of the
%    material contained in \bbox. #1 = macro constructs a #1pt
%    separation around the material in \bbox
\ifdvipsone
%
% currentpoint and box dimensions are in TeX's scaled pts. Combine
% these first, then convert to printers points using TeXtoPDF
\def\FitRbboxB#1{ % Uses \bbox
   currentpoint 2 copy \the\hdpth\space add TeXtoPDF
   neg PageHeight add 72 sub #1\space sub      % y1
   exch TeXtoPDF 72 add #1\space sub exch      % x1
   4 2 roll exch \the\hwdth\space add
   TeXtoPDF 72 add #1\space add exch           % x2
   \the\hhght\space sub TeXtoPDF neg
   PageHeight add 72 sub #1\space add}         % y2
\else  % dvips
%
% currentpoint is in "pixels" convert to pts with DvipstoPDF. Box
% dimensions are in scaled points, convert to  pts with TeXtoPDF
```

```
\def\FitRbboxB#1{% Uses \bbox
  currentpoint 2 copy DvipstoPDF \the\hdpth\space TeXtoPDF add
  neg vsize add 72 sub #1\space sub exch      % y1
  DvipstoPDF 72 add #1\space sub exch         % x1
  4 2 roll exch DvipstoPDF \the\hwdth\space
  TeXtoPDF add 72 add #1\space add exch       % x2
  DvipstoPDF \the\hhght \space TeXtoPDF sub
  neg vsize add 72 sub #1\space add}          % y2
\fi
```

Here's another illustration of this macro now that you've seen it.

Restore

Isaac Newton (1642–17270) Newton was born in Woolsthorpe, England. In 1661 he entered Trinity College in Cambridge and was taught by Isaac Barrow, a well-know mathematician and teacher. Plague closed Trinity College, and Newton returned to Woolsthorpe. In the two years that followed (1665–1666), Newton discovered calculus and recognized the underlying principles of planetary motion and gravity.

The details of these links are much the same as the previous example. "Isaac Newton" is a hypertext jump to his own picture:

```
\htxt{\hg{Issac Newton}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Dest /GoToNewton
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Just *before* the picture insertion, there is a destination mark:

```
\special{ps: %
[ /Dest /GoToNewton
  /View [ /FitR \FitRbboxB4 ]
  /DEST pdfmark}\unhbox\bbox
```

In this case, `\bbox` contains the `tif` picture of Newton with the hypertext "Restore" below it; and we do a `\unhbox\bbox` to actually show the material. All this material is put into `\bbox` so its dimensions can be taken and used by the framing macro `\FitRbboxB`. (Here, I decided to have a 4pt separation around the material.)

How the "Restore" link works is discussed in SECTION 1.9.

I use `tif` files because Y&Y has a special '`\special`' `tif` insertion:

`\special{insertimage: <tif file> <width> <height>}`.

This gives me a preview for the `dvi` file, using DVIWindo, as well as for the `pdf` file. Any of the standard figure insertion packages for TEX,

LaTeX, and LaTeX2e can be used. The important point is to place your destination mark in the proper location and to get the dimensions of the bounding box of the figure. Example 1.24. ■

1.9. Named Actions

Named actions, first mentioned in [pdfs, p. 110], are used by Adobe to execute *menu functions*. In Thomas Merz's book, Web Publishing with Acrobat/PDF, [webpub], a fairly complete catalog of all menu functions is given. This listing is available through the Web in his Pdfmark Primer.

I shall content myself with giving the proper syntax and a few examples. Named actions require the custom action form of the link:

```
/Action << /Subtype /Named /N <name> >>
```

where `<name>` is the (correct) name of the menu function to be performed. The key word `/Subtype` can be shortened to `/S`.

One example of named actions already seen can be found in line (2) of Example 1.9. Here's another example of a link already seen.

The following link has the same effect as View > Go Back from within the Acrobat Reader:

▶ Restore

```
\htxt{\hg{Restore}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /Named /N /GoBack >>          % Named Action
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

This example will go to the previous view—probably the previous page. In the Newton Example, you zoomed into the picture of Newton, then clicked on "Restore" to go to previous view. It worked in well in that case. (unless you clicked on "Restore" before clicking on "Isaac Newton").

One last example to illustrate named actions.

▶ File > Preferences > Weblink...

```
{\tt File > Preferences > \htxt{\hb{Weblink...}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 0 ]
  /Action << /S /Named /N /Weblink:Prefs >>  % Named Action
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox}
```

Here is a short list of named actions. The first four are the ones listed in [pdfs, p. 110].

- `/N /FirstPage` to go to the first page;
- `/N /LastPage` to go to the last page;
- `/N /PrevPage` to go to the previous page (the single left arrow on the menu bar);
- `/N /NextPage` to go to the next page (the single right arrow on the menu bar);
- `/N /GoBack` to go the to previous view (the left double arrow on the menu bar);
- `/N /GoForward` to go to the next view (the right double arrow on the menu bar).

See the [primer] for a "complete" listing of all named actions—as current as Acrobat 3.01, excluding Forms 3.5.

1.10. Multiple Actions

[pdfs, p. 101] states that a series of actions can be performed using the `/Next` key. The value of the `/Next` key is a dictionary or an array. It took me quite a while to figure how to use this key so it would work.

Having finally mastered the `/Next` key, I used it in Algeboard and Giants of Calculus to create links and form buttons with multiple actions—mostly in conjunction with manipulating form fields.

The following example performs three actions: (1) Changes the view on the current page; (2) jumps to a named destination page containing the mark `TargetPage`; and (3) performs a named action—view weblink preferences.

▶ Click Here

```
\htxt{\hb{Click Here}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action << /S /GoTo /D [ {ThisPage} /XYZ null null 4 ]
    /Next << /S /GoTo /D (TargetPage)
    /Next << /S /Named /N /Weblink:Prefs
  >> >> >>
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

Example Notes: The actions are nested; part of each action is to initiate the next action.

- This form of the `/Next` key is a dictionary.
- The most interesting and useful applications of multiple actions, I feel, are ones that process form fields. In the next section, the topic of defining form fields using `pdfmarks` is taken up. At that time, additional examples of multiple actions will be given. Until then, this one must suffice. ■

1.11. Linking Icons and Pictures

No treatment of hypertext would be complete without some mention of linking to objects other than words or phrases. In this section methods of creating links to icons and pictures are discussed.

- **Using Fonts and Rules**

Let's create a simple button "icon" out of what is available from TEX. This particular icon is used in my tutorial to guide the reader from one main page to another. (See the first page of this article.)

The rectangular icon (or button) to the left was constructed from the `\Uparrow` (⇑) character available from the Computer Modern Font Set. The background is a `\vrule` that has been colored a `\nicegray`—the arrow is a `\niceblue`. The details follow.

First a few preliminary macro definitions.

```
%
% Here's a macro for putting separation around the material #5.
% Horizontal: #1 and #4.  When #1=#4, #5 is centered horizontally.
% Vertical: #2 and #3. When #2=#3, #5 is centered vertically.
%   By playing around with #1--#4, you can move #5 anywhere within the
% bounding invisible rectangle.  Use a colored rule to throw in a
% colored background.
\def\surroundit[#1,#2,#3,#4]#5{%
```

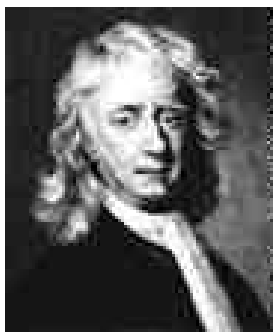
```
  \vbox{\hbox{\kern#1pt\vbox{\kern#2pt#5\kern#3pt}\kern#4pt}}}
%
% A few nice colors
\def\niceblue{0 0 .5} \def\nicegray{.753 .753 .753}
\setbox0=\hbox{\surroundit[6,6,6,6]     % 6 pt surround
  {\hbox{\pushcolor\niceblue$\bigl\Uparrow$\popcolor}}}%
\htxt{%                                 % puts material in \bbox
\pushcolor\nicegray                     % push gray (for background)
\rlap{\vrule width \wd0 height \ht0}%   % color rule makes background
\unhbox0                                % unbox the uparrow
\popcolor}\special{ps: %                % pop color (nicegray)
[ /Rect \Rect /Border [ 0 0 0 ]
  /Action /GoTo /Page 1
  /Subtype /Link
/ANN pdfmark}\unhbox\bbox
```

The details of the paragraph indentation are found below.

```
\setbox0=\hbox{\surroundit[6,6,6,6]
  {\hbox{\pushcolor\niceblue$\bigl\Uparrow$\popcolor}}}%
\setbox2=\hbox{\htxt{\pushcolor\nicegray
  \rlap{\vrule width \wd0 height \ht0}%
\unhbox0\popcolor}%
\special{ps: [ /Rect \Rect /Border [ 0 0 0 ] /Action /GoTo
/Page 1 /Subtype /Link /ANN pdfmark}\unhbox\bbox}%
\noindent\hangindent=\wd2 \advance\hangindent3pt\hangafter=-2%
```

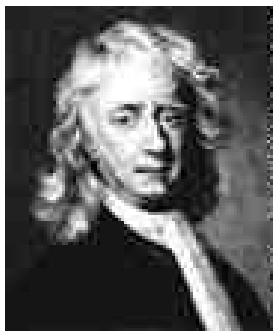
```
\smash{\lower1.2\baselineskip\hbox to0pt{\kern-\hangindent
\unhbox2\hss}}The rectangular icon (or button) to the left
was constructed from...
```

- **Using Tiff images**



Here is Isaac Newton again. This time, he is acting as a link button. Press on him and jump to a destination within this document. We just enclose the `tif` image in a `\vbox` and put it into the macro `\htxt`. This time, the details of the paragraph indentation are included for your study; put everything into an `\hbox` so the dimensions can be measured for `\hangindent`.

```
\setbox0=\hbox{\htxt{\vbox{%
\showtiff{newton.tif}{45pt}{90}{109}}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ] /H /N  % no highlighting
  /Action << /S /GoTo /D (NewtonExmpl) >>
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
} %\setbox0
\noindent\hangindent=\wd0 \advance\hangindent3pt\hangafter=-5 %
\smash{\lower4\baselineskip\hbox to0pt{%
\kern-\hangindent\unhbox0\hss}}Here is Isaac Newton again.  This time,
he is acting as a link button. Press on him and jump to a destination
within this document...
```



Perhaps it is not obvious that old Isaac is a hypertext link. In this case, The `/Color` key can be used to place a colorful border around Isaac (recall, `/Border [0 0 1]` is the default border); highlighting can also be changed, say, to outlining `/H /O`. Any action can be attached to an icon or picture—even launching an application, for example.

```
\setbox0=\hbox{\htxt{\vbox{%
\showtiff{newton.tif}{45pt}{90}{109}}}\special{ps: %
[ /Rect \Rect
  /Border [ 0 0 2 PDFtoTeX ] % dvips: use ‘2 PDFtoDvips’ instead
  /Color [ 0 1 0 ]           % green border,
  /H /O                      % outline highlighting
  /Action << /S /GoTo /D (NewtonExmpl) >>
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
} %\setbox0
```

- **Using EPS Images**

There is a lot of `gif` icons and other graphics available over the Web, and if you find something interesting, it's only natural to incorporate it into an electronic document of your own.

To the left is a push button icon—originally a `gif` file—that I downloaded from some `html` file somewhere in the universe. Acrobat 3.0$^+$ comes with Photoshop Lite so I loaded the `gif` file, `home.gif`, into Photoshop Lite, modified the text appearing on the button a little, then saved it as an `eps` file, `home.eps`.

Photoshop Lite saves an `eps` file with a `tiff` preview. This is convenient for the Y&Y System since they have a built in `\special` for viewing a `tiff` file from within DVIWindo. What you see is the `tiff` preview of the file `home.eps`.

The code used to insert the above icon button, without the paragraph indentation code, is

```
\htxt{\vbox{\showtiff{home.eps}{50pt}{100}{25}}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ] /H /I   % Invert highlighting
  /Action << /S /GoTo /D [ {Page1} /XYZ null null null ] >>
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

You'll notice that I have set the highlighting to invert: `/H /I`. It doesn't look that great, but it's better than some of the alternatives. Perhaps a better choice would have been `/H /N`, no highlighting.

DVIPSONE We can insert the same icon using `epsfsafe.tex`:


```
\begingroup
\input epsfsafe \epsfxsize=50pt
\rtr\htxt{\lower2pt\hbox{\epsfbox{home.eps}}}\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ] /H /I   % Invert highlighting
  /Action << /S /GoTo /D [ {Page1} /XYZ null null null ] >>
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
\endgroup
```

Code Notes: DVIWindo uses the `tiff` preview but when DVIPSONE is used to convert the `dvi` file to a `ps` file, the `eps` image is used instead of the `tiff` preview. *Very nice, . . . and easy to use too*!

- I've enclosed the code in a `\begingroup` . . . `\endgroup` pair because this is the only instance of using `epsfsafe.tex`.
- `epsfsafe.tex`, distributed by Y&Y, is a variation on the standard distribution file `epsf.tex`; however, the Y&Y version detects and uses a `tiff` preview if present in the `eps` file. ■

Dvips. Speaking of `epsf.tex`, the same effects can be obtained by DVIPS as well, though there will not be a `tiff` preview available for the `dvi` previewer.

Initially, I had problems getting my MikTeX system to show the `eps` file properly. DVIPS gave me the error message:

```
! premature end of file in binary section
```

and the file would get truncated after the image insertion.

I finally hit on the solution: Photoshop has the option of saving the `EPS` image data in ASCII or binary; I had saved in ASCII format, which DVIPSONE liked but which DVIPS choked on. Switching over to binary got rid of error message.

With or without the `tiff` preview present in `home.eps`, `epsf.tex` had trouble finding the bounding box and DVIPS had trouble displaying the image properly (the bounding box was taken as the dimensions of the entire page). The macro `\epsfbox{home.eps}` was *not sufficient*—as it had been for DVISPONE—to get the `eps` image to appear properly after distillation.

It was necessary to specify explicitly the dimensions of bounding box, which were 100pt wide and 25pt high, using the macro

```
\epsfbox[0 0 100 25]{home.eps}.
```

Finally, everything worked as advertised!

For DVIPS, the same icon can be inserted, though it is not done here because I am using DVIPSONE, using the code:

```
%
% Code for EPS insertion using epsf.tex
%
\input epsf \epsfframetrue \epsfxsize=50pt
\htxt{\epsfbox[0 0 100 25]{home.eps}}%
\special{ps: %
[ /Rect \Rect /Border [ 0 0 0 ] /H /I   % Invert highlighting
  /Action << /S /GoTo /D [ 0 /XYZ null null null ] >>
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

This code works fine on my MikTeX system.

1.12. Jumping to Local Files

As long as you reference files using relative paths, the links created will work on the WEB or on your local file system.

If you are constructing a system of `pdf` files that are meant to be strictly used on a local hard drive, it may occasionally be necessary to use absolute paths. Table 1 has examples of the `PDF` file specification convention. (The table is an exact copy of the one that appears in [pdfs, p. 137], reproduced here for your convenience.)

Table 1. Examples of File Specification

System	System-dependent page	Written as ...
Mac	Macintosh HD:PDFDocs:spec.pdf	(/Macintosh HD/PDFDocs/spec.pdf)
DOS	\pdfdocs\spec.pdf (no drive)	(//pdfdocs/spec.pdf)
DOS	r:\pdfdocs\spec.pdf	(/r/pdfdocs/spec.pdf)
DOS	pcadobe/eng:\pdfdocs\spec.pdf	(/pcadobe/eng/pdfdocs/spec.pdf)
UNIX	/user/fred/pdfdocs/spec.pdf	(/user/fred/pdfdocs/spec.pdf)
UNIX	pdfdocs/spec.pdf (relative)	(pdfdocs/spec.pdf)

System Dependent External File Keys. Acrobat provides several external file keys designed to be used with the common Operating Systems listed above; they are `/DOSFile`, `/MacFile` and `UnixFile`. Each of these keys takes a *string* as its value that specifies the, possibly absolute, path to the file (including filename).

Important: The `/File` key is *required* to be present for all `/GoToR` actions; when one of these three keys is present, however, Acrobat *ignores the* `/File` key.

Both of the following links would work (though, they are non-working examples).

Using /DOSFile

```
\htxt{Click Here}\special{ps: %
[ /Rect \Rect
  /Action /GoToR
  /Dest /MyMark
  /File ()
  /DOSFile (//temp/dps.pdf)
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

Using /File

```
\htxt{Click Here}\special{ps: %
[ /Rect \Rect
  /Action /GoToR
  /Dest /MyMark
  /File (c:/temp/dps.pdf)
  /Subtype /Link
  /ANN pdfmark}\unhbox\bbox
```

Example Notes: I have left the value of the `/File` key as null; that's all right.

- In the left-hand example, `/DOSFile (c:/temp/dps.pdf)` works; however, `/DOSFile (/temp/dps.pdf)` *does not work.*
- For the right-hand example, `/File (//temp/dps.pdf)` works, but the key-value pair `/File (/temp/dps.pdf)` *does not.*
- *Summary*: Acrobat does not like driveless absolute paths; in this case the root directory must be referred to by the double slash '`//`'.
- At least on **Win95** machines, the `/File` key works well with *relative paths* and with *absolute paths* with the *drive letter specified.*
- The need for these special keys may be rather infrequent; still, each OS has its little idiosyncrasies ■

1.13. Hypertext using Y&Y

The TEX system developed by Y&Y has hypertext capabilities built into their `dvi` previewer, DVIWindo. Within DVIWindo, these links are fully functional; when a `dvi` file containing hypertext links is "printed" to a PostScript file using DVIPSONE, these hypertext links are automatically converted over to the appropriate Acrobat `pdfmarks`.

Typically, when you "roll your own" hypertext links using the techniques developed above, they will not work from within DVIWindo, but they will work with Acrobat Reader—after you convert to `pdf` using the Acrobat Distiller.

The reason for this is the Y&Y uses certain special `\special`'s to define destinations, and to define hypertext links to these destinations. This enables DVIWindo to organize these destinations and links into a reference table that can be used by the DVIWindo.

There are fundamentally two specials that Y&Y defines for creating hypertext links and destinations:

- **Destinations.** To create a named destination, or 'mark', use the special:

```
\special{mark: <markname>}
```
(8)

where, `<markname>` is the name of the destination.

- **Links.** A link is defined with a 'button' special:

  ```
  \special{button: <width> <height> <action>}
  ```
  (9)

  where `<width>` and `<height>` are the dimensions of the bounding rectangle, as measured in TEX's scaled points. (*Recall*: There are 65,536 scaled points per printer's point.) There are several possible meanings for `<action>`, as will be illustrated in the examples below.

Additional Notes: To create a link, the button text (or icon) is placed in a box, its dimensions taken and used for `<width>` and `<height>`, the text is then unboxed just after `\special` to reveal the text. ∎

The macro `\htxt` already developed serves perfectly adequately for defining Y&Y hypertext jumps. The dimensions of the argument of `\htxt` are transferred to the count registers `\hwdth` and `\hhgth`; these are exactly the `<width>` and `<height>` required for Y&Y's button special.

Example 1.25. Jump to a Page. In order to jump to a particular page number within the current `dvi/pdf` document, use the following:

▶ Jump to a Page 5 in this Document

```
\htxt{\hg{Jump to a Page 5 in this Document}}\special{button:
\the\hwdth\space\the\hhght\space
page: 5}\unhbox\bbox
```

Example Notes: In this case, '`<action>` = `page: 5`'. This example is equivalent to Example 1.8.

For jumping to a particular page in another document (`dvi` or `pdf`), we take '`<action>` = `file: -p=<n> <filename>`', where '`<n>`' is the page number, and '`<filename>`' is the name of the file, without extension.

The next example will work correctly if you have dvipsone 2.1.3 or higher.

▶ Jump to a Page 2 in another Document

```
\htxt{\hg{Jump to a Page 2 in another Document}}\special{button:
\the\hwdth\space\the\hhght\space
file: -p=2 examples/sample}\unhbox\bbox
```

The above example duplicates Example 1.13—that link will not work in dviWindo . . . this one will. Example 1.25. ■

Example 1.26. Open a File. Opening a file means jumping to page one of another document. The next example performs the same task as EXAMPLE 1.12, note that there is no page specification:

▶ Go to a File

```
\htxt{\hbr{Go to a File}}\special{button: %
\the\hwdth\space\the\hhght\space
file: examples/sample}\unhbox\bbox
```

Example 1.26. ■

Example 1.27. Define a Y&Y Mark. In order for the hypertext jumps to work properly in DVIWindo, you must use Y&Y's special '`\special`'. The correct syntax is given in line (8).

The following defines a mark that both DVIWindo and the Acrobat Reader will recognize: `\special{mark: yyThisPage}` **Note:** There is actually a mark here, though it is invisible to the viewer.

Alternately, you can define a named destination using the methods enumerated in EXAMPLE 1.10. This will create a named destination in the `pdf` file (after you Distill), but the destination, or mark, will not be unknown to DVIWindo.

Example 1.27. ■

Example 1.28. Jump to a Mark. This is the same as jumping to a named destination in Acrobat jargon.

Take '`<action> = <markname>`' to get a link to a destination that will work in both DVIWindo and Acrobat Reader—after you Distill, of course.

▶ Jump to Y&Y Mark within Document

```
\htxt{\hg{Jump to Y\&Y Mark within Document}}\special{button:
\the\hwdth\space\the\hhght\space GoYandY}\unhbox\bbox
```

Should you want to jump to a destination mark in another file, in this case, take '`<action> = <markname>, file: <filename>`'. **Note:** Be sure to separate the '`<markname>`' from the '`file: <filename>`' by a *comma*.

▶ Jump to Y&Y Mark in another Document

```
\htxt{\hbr{Jump to Y\&Y Mark in another Document}}\special{button:
\the\hwdth\space\the\hhght\space
GetYandY, file: examples/sample}\unhbox\bbox
```

Example Notes: For these links to work properly in DVIWindo, it is important that the designation mark be defined using Y&Y's special—as described in Example 1.27. Example 1.28. ■

Example 1.29. Jump to an URL. The current version of DVIWINDO does not support jumping to destinations on the Web; however, you can create URL's what will work when you distill.

Take `<action>` = valid URL. For example,

▶ Go to e-Calculus

```
\htxt{\hbr{Go to \eCalculus}}\special{button:
\the\hwdth\space\the\hhght\space
http://www.math.uakron.edu/\noexpand~dpstory/e-calculus.html}%
\unhbox\bbox
```

You can jump to an 'anchor' in an `html` document, or to a 'named destination' in a `pdf` document by post fixing '`#<anchor/destname>`'.

The following is the equivalent of Example 1.15 and would get expanded into the syntax of Example 1.15 when distilled:

▶ Click Here

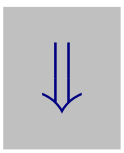
```
\htxt{\hbr{Click Here}}\special{button:
\the\hwdth\space\the\hhght\space
http://www.math.uakron.edu/\noexpand~dpstory/%
  tutorial/pdfmarks/examples/sample.pdf\#Target}\unhbox\bbox
```

Example 1.29. ■

Since I use the Y&Y System, all of the cross-reference links of this article were created using Y&Y's link and mark specials.

Obviously, sophisticated macros can be constructed to implement these specials to help cut down on the amount of typing. In these

notes, I didn't see a need to publish the particular macros I use myself in my tutorials. *Creation of a set of macros is left as an exercise for the reader—that's you.* 𝔇𝔓



This ends the hypertext links portion of this article. Click on the down arrow to continue with the article on forms.

Destination Page

You have reached the *within document* target page. Let's plant a target destination right here:

```
\special{ps: %
[ /Dest /TargetPage
  /DEST pdfmark}
```

I've included the mark, `/TargetPagewithFit`, that controls the view of this page—when we jump to this page using this mark, of course.

```
\special{ps: %
[ /Dest /TargetPagewithFit
  /View [ /Fit ]
  /DEST pdfmark}
```

Y&Y Destination Page

When using the Y&Y System, a mark is "planted" using the syntax: `\special{mark: GoYandY}` The actual mark is invisible to the viewer—that's you.

The 'neat' thing about Y&Y's hypertext scheme is that when you jump to a mark (from within DVIWindo) the mouse cursor points to the exact location of the mark. This feature is very useful for directing the user's attention to a particular word, phrase, or equation. The Adobe Reader simply loads the correct page.[4]

[4] Actually, more can be said. When the page has not been `/Fit` to the window, the Reader will show that portion of the partial page that contains the destination mark. For example, if the destination mark is on the bottom third of the page, the Reader will try to show the lowest third of the page. The Reader does not move the cursor during a jump though.

This is the left column in this two column page. On this page is the destination mark:

```
\special{ps: %
[ /Dest /TwoColPageL
  /View [ /FitV 108 ]
  /DEST pdfmark}
```

This mark gives the `/FitV` key word a value equal to the x-offset of the *left* edge of the cropped page. (We cropped 108pts off the left side of the original 8.5" x 11" page.)

This is the right column in this two column page. On this page is the destination mark:

```
\special{ps: %
[ /Dest /TwoColPageR
  /View [ /FitV 108 396 add ]
  /DEST pdfmark}
```

This mark gives the `/FitV` key word a value equal to the x-offset of the *right* edge of the cropped page. (We cropped 108pts off the left side of the original 8.5" x 11" page; the cropped page is 396pts wide.)

The marks themselves can appear most anywhere on the page. They don't have to be in the particular columns themselves.

Bibliography

[dvips] The Official Dvips Home Page, Radical Eye Software http://www.radicaleye.com/dvips.html

[miktex] The MikTEXProject Page, Freeware TEX for Win32 http://www.inx.de/~cschenk/miktex/

[pdfm] pdfmark Reference Manual, Technical Note #5150, by Tim Bienz and Gary Staas, Adobe Systems Incorporated, October 24, 1996.

[pdfs] Portable Document Format Reference Manual, Version 1.2, by Tim Bienz, Richard Cohn, and James R. Meehan, Adobe Systems Incorporated, November 12, 1996.

[primer] The Pdfmark Primer, Thomas Merz, freely available over Web, http://www.ifconnection.de/~tm/, 1998

[tex] *The TEXbook*, Donald Knuth, Addison-Wesley Publishing Co., June 1992.

[y&y] The Y&Y Inc. Homepage, http://www.yandy.com/

[webpub] *Web Publishing with Acrobat/PDF*, Thomas Merz, Springer -Verlag, 1998