Enhancing the Text in Text Fields using Pdfmarks

by

D. P. Story

1. Introduction

This article is primarily concerned with enhancing the appearance of the text of a text field beyond what is already documented in **Section 6** of the *pdfmark Reference Manual* [3]. A good general reference to pdfmarks is *The Pdfmark Primer* by Thomas Merz [5]; see also the article by D. P. Story entitled *Pdfmarks: Links and Forms* [6] for many technical details about creating and enhancing the appearance of the various form annotations in a PDF document.

Let's begin by illustrating what is meant by a text field. Below is an example of a standard text field created using pdfmarks.

Enter your name here

Standard Text Box

```
[ /Rect [ xll yll xur yur ]
  /T (YourName)
  /FT /Tx  /F 4
  /BS << /W 2 /S /S >>
  /MK << /BC [ 0.9647 0.6274 0.7882 ]
         /BG [.7529 .7529 .7529] >>
  /DA (/Helv 10 Tf 0 0 1 rg)
  /DV (Enter your name here)
/Subtype /Widget /ANN pdfmark
```

What is meant by enhancing the text? Certainly, we can change fonts to any of the Base 13, and we can change the style, color and size as well. Beyond these already documented appearances, what is possible? Here's an example: **Push**

Enhanced Text Box

Edit the **Enhanced Text Box** to see that the text entered is again *stroked and filled* as above.

Important Reminder. For the interactive examples that follow, the data is not "committed" until the user exits the field by clicking outside the field, or by tabbing to the next field, or by pressing enter.

I have created a Valentine's Day Card to illustrate all of the new enhancements of the text appearance within the context of a nice application.

Before continuing with this article, can you figure out how it is done?

2. Referencing the PDF Reference Manual

Form features of the Portable Document Format (PDF) were introduced in Version 1.2, as documented in the *Portable Document Format Reference Manual*, dated March 1, 1996 [1] (the most recent revision of Version 1.2 is that of November 12, 1996 [2]). Though this manual is primarily intended for application developers who wish to produce PDF files directly, it contains much valuable information for anyone having a desire to access the powerful annotation features of PDF through the pdfmark.

The enhancements come from modifying the **DA** key. The standard form for the **DA** key, as given in the Standard Text Box, is

```
/DA (/Helv 10 Tf 0 0 1 rg)
```

The **Tf** operator sets the font and size [2, p. 234] and the **rg** operator sets the rgb fill color [2, p.221] of the text in the field.

In §6.15.6, **Fields comprising variable text**, of [2, p. 125], we find an initial description of the **DA** operator:

> (Required, inheritable) Default appearance string. This string contains a sequence of valid page-content graphics or text-state operators that determine such appearance properties as text color, text size, etc.

On page 212 of [2] we find a list of valid text-state operators:

- General Graphics State: **d, gs, i, j, J, M, w**
- Color: **g, G, k, K, rg, RG, sc, SC, scn, SCN, cs, CS**
- Text State: **Tc, Tf, TL, Tr, Ts, Tw, Tz**
- Text String (Painting): **Tj, TJ, ', "**
- Text Positioning: **Td, TD, Tm, T***

Of particular interest to this article are the legal Color and Text State operators.

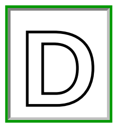
3. Text State Operators

In this section we examine the Text State operators: **Tr**, **Tc**, **Tw**, and **Tz**. The Color operator **RG** will also be used. These operators are illustrated using text fields, but they work for any variable text field. (Variable text fields include the **Text Field**, the **List Field**, the **Pop-up List** and the **Combo Box**.)

• The **Tr** Operator

The **Tr** operator is used to set the *text rendering mode*. It takes one argument, numbered 0 through 7. Only the first three modes are useful at this time: `0 Tr` (fill text, the default); `1 Tr` (stroke text); `2 Tr` (stroke and fill text).

Example 1. Stroke Text (rendering mode 1).



Stroke Text

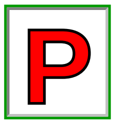
```
[ /Rect [ xll yll xur yur ]
  /T (D) /FT /Tx /MaxLen 1
  /F 4 /BS << /S /I >>  /Q 1
  /MK << /BC [ 0 .6 0 ] >>
  /DA (1 Tr /HeBo 40 Tf 1 0 0 rg)
  /DV (D)
/Subtype /Widget /ANN pdfmark
```

Notice the default appearance string

```
/DA (1 Tr /HeBo 40 Tf 1 0 0 rg)
```

calls for *rendering mode* 1 (stroke), Helvetica-Bold at 40 points, and a red text, ‘`1 0 0 rg`’. The red does not appear because we are not filling the text. Without the ‘`1 Tr`’, rendering mode 0 would be in effect (fill), the letter would be painted red. ■

Example 2. Stroke and Fill Text (rendering mode 2).



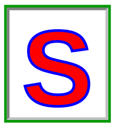
Stroke and Fill Text

```
[ /Rect [ xll yll xur yur ]
  /T (P)
  /FT /Tx /MaxLen 1
  /F 4 /BS << /S /I >>  /Q 1
  /MK << /BC [ 0 .6 0 ] >>
  /DA (2 Tr /HeBo 40 Tf 1 0 0 rg)
  /DV (P)
/Subtype /Widget /ANN pdfmark
```

This is the same as example as Example 1, except now the text is filled and stroked, ‘`2 Tr`’. Since the text is filled, the **rg** operator paints the interior red, ‘`1 0 0 rg`’. ■

With respect to stroking the text, the Color operator **RG** comes in handy; **RG** sets the color space to **DefineRGB**, and sets the color to use for stroking paths [2, p. 221]. (Quite similar in use to the more familiar **rg**.)

Example 3. Stroke and Fill Text (rendering mode 2), color the stroke.



Color Stroke and Fill Text

```
[ /Rect [ xll yll xur yur ]
  /T (S)
  /FT /Tx /MaxLen 1
  /F 4 /BS << /S /I >>  /Q 1
  /MK << /BC [ 0 .6 0 ] >>
  /DA (2 Tr /HeBo 40 Tf 0 0 1 RG 1 0 0 rg)
  /DV (S)
/Subtype /Widget /ANN pdfmark
```

The ‘`2 Tr`’ in the default appearance string defines *rendering mode* 2, stroking and filling the text; the insertion of ‘`0 0 1 RG`’, this calls for the stroked text to be colored blue. ■

• The **Tc** and **Tw** Operators

The **Tc** operator takes a single argument, the argument is the amount of space after a character, while **Tw** sets the word spacing with its argument. **Tc** and **Tw** can be used to obtain an interesting spread out look.

Example 4. Set Character and Wording Spacing.

E n t e r   y o u r   n a m e   h e r e

Set Character and Word Spacing

```
[ /Rect [ xll yll xur yur ]
  /T (YourName)  /FT /Tx
  /F 4 /BS << /S /I >>
  /MK << /BC [ 0 .6 0 ] >>
  /DA (6 Tw 6 Tc /HeBo 12 Tf 1 0 0 rg)
  /DV (Enter your name here)
/Subtype /Widget /ANN pdfmark
```

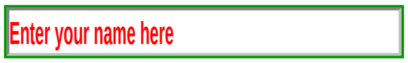
Here the word spacing, ‘`6 Tw`’, is set to 6 text-space units, as is the character spacing, ‘`6 Tc`’.

Remark: Had you entered your name in the Standard Text Box, your name would have been transferred to this text box because the two text boxes share the same title, '`/T (YourName)`'. ■

• The **Tz** Operator

The operator **Tz** sets the horizontal spacing, its argument is a number expressed as a percent of the normal scaling. This operator can be used to "compress" the text.

EXAMPLE 5. Change the horizontal scaling.



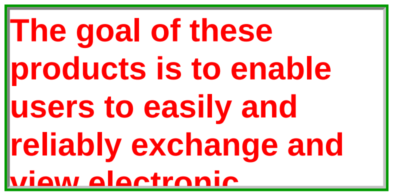
Enter your name here

Color Stroke and Fill Text

```
[ /Rect [ xll yll xur yur ]
  /T (YourName) /FT /Tx
  /F 4 /BS << /S /I >>
  /MK << /BC [ 0 .6 0 ] >>
  /DA (50 Tz /HeBo 12 Tf 1 0 0 rg)
  /DV (Enter your name here)
/Subtype /Widget /ANN pdfmark
```

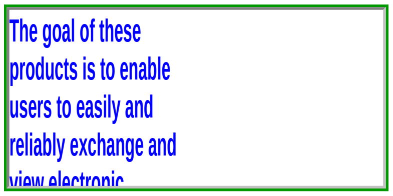
The horizontal scaling parameter is set to 50, '`50 Tz`'; thus, the scaling is 50% of normal. ■

Here is an interesting problem, discovered in the course of creating the Valentine's Day Card.



The goal of these products is to enable users to easily and reliably exchange and view electronic

Multiline Text Box Horizontal Scale 100%



The goal of these products is to enable users to easily and reliably exchange and view electronic

Multiline Text Box Horizontal Scale 50%

```
[ /Rect [ xll yll xur yur ]
  /T (InputMessage)
  /FT /Tx /F 4 /Ff 4096
  /BS << /S /I >>
  /MK << /BC [ 0 .6 0 ] >>
  /DA (/HeBo 12 Tf 1 0 0 rg)
  /DV {datastring}
  /AA  {AAjavascript}
/Subtype /Widget /ANN pdfmark
```

```
[ /Rect [ xll yll xur yur ]
  /T (OutputScaledMessage)
  /FT /Tx /F 4 /Ff 4097
  /BS << /S /I >>
  /MK << /BC [ 0 .6 0 ] >>
  /DA (50 Tz /HeBo 12 Tf 0 0 1 rg)
  /DV {datastring}
  /Subtype /Widget
/ANN pdfmark
```

Note the use of the operator **Tz** in the default appearance of the code on the right. Within **DA** is '`50 Tz`' indicating a 50% horizontal scaling of the text.

A **Cos object**, [3, p. 21] was used to store the default message, '`/DV {datastring}`'. I did this for two reasons: (1) stream data stored in this way is always compressed using a lossless method (either LZW or ZIP), [3, p. 23]; (2) I couldn't fit the code in with a two column format. I stored the additional actions key-value pair this way as well, '`/AA {AAjavascript}`'. The following is the **Cos object** code.

```
[ /_objdef {datastring} /type /stream /OBJ pdfmark
[ {datastring} (The goal of these products is to enable users to easily
    and reliably exchange and view electronic documents independent
    of the environment in which they were created.) /PUT pdfmark
[ {datastring} /CLOSE pdfmark
[ /_objdef {AAjavascript} /type /dict /OBJ pdfmark
[ {AAjavascript} << /V << /S /JavaScript
  /JS (this.getField("OutputScaledMessage").value=event.value;) >> >>
/PUT pdfmark
```

There are a number of comments to be made about this example. The interesting point that caught my attention is that when transferring data using JavaScript, the *line breaks* are preserved. In fact, I designed the Valentine's Day Card in such a way to mask over this potential problem.

I used JavaScript to transfer the data from one text field to the other because I wanted, in the Valentine's Day Card application, the target field to be *read-only*. For fields with the same title, if one is read-only, so is the other. The JavaScript solved the problem of inputting into one field, and outputting into another read-only field.

The JavaScript is really not the problem. The same problem occurs when you remove the read-only specification and make each field have the same title. The line breaking algorithm uses the width of the bounding box; a box that is 300 points wide, for example, is computed to be 150 points wide at 50% horizontal scaling.

What if you wanted to rescale text by 50%, but wanted new line breaks to fit into another region that is 300 points wide? The answer is to create a text box 600 points wide! ■

Other text-state operators are valid as well: some don't do anything, for others, no significant application can be imagined. Additional features, not yet known, may be discovered by a very close reading of the *Portable Document Reference Manual*.

References

[1] Adobe Systems Incorporated, *Portable Document Format Reference Manual*, Version 1.2, March 1, 1996.

[2] Adobe Systems Incorporated, *Portable Document Format Reference Manual*, Version 1.2, November 12, 1996.

[3] Adobe Systems Incorporated, *pdfmark Reference Manual*, Technical Note #5150, October 24, 1996.

[4] Thomas Merz, *Web Publishing with Acrobat/PDF*, Springer-Verlag Berlin Heidelberg New York, 1998, ISBN 3-540-63762-1.

[5] Thomas Merz, *The Pdfmark Primer*, freely available over the Web, `http://www.ifconnection.de/~tm/`, also available at `http://www.pdfzone.com/pdfs/merzprimer.pdf` (706 kb) 1998.

[6] D. P. Story, *Pdfmarks: Links and Forms*, freely available over the Web, `http://www.math.uakron.edu/~dpstory/acrotex.html`, 1998.