www.AcroTeX.net

AcroTEX.Net

The fitr Package Defining and Jumping to a Rectangular Destination

D. P. Story

www.acrotex.net
Version 1.3.2, 2020/07/09

The fitr Package

Table of Contents

1. Introduction

This package is an implementation of the **FitR** view-type destination. The *PDF Reference* describes **FitR** as,

> Display the page designated by page, with its contents magnified just enough to fit the rectangle specified by the coordinates *left*, *bottom*, *right*, and *top* entirely within the window both horizontally and vertically.

The package supports the dvips, pdflatex, lualatex, xelatex, dvipdfm, and dvipdfmx applications.

The only required packages are the eforms package (dated 2012/06/20 or later), which is part of the AeB Bundle, and collectbox by Martin Scharrer, more on this package later, and the ubiquitous xcolor.[1]

Motivation. This package was developed in response to a user of the AeB Bundle who was interested in developing documents for students with *low vision*; the idea is to magnify regions of the document so the student can read more comfortably.

Demonstration files. The demonstration files are `fitr_demo.tex` which illustrates the package and some special methods for people with low vision, and `fitr_minimal.tex`, which is the same demo file with any extra packages stripped out.

What's New for Version 1.3.2 (2020/07/09) The package is extended to support the PDF creator lualatex. All common workflows are now automatically detected: pdflatex, lualatex, and xelatex. If no driver is specified and none is automatically detected, dvips is automatically selected by default. The other drivers dvipdfm and dvipdfmx need to be explicitly specified as the driver (⟨*driver*⟩). The ⟨*driver*⟩ option is described in the next section. Also, there are several new options, one of which is `gonative`; when this option is selected, no field or document JavaScript is used. For the `gonative` option, the workflow dvips -> ps2pdf is also supported because there is no document JavaScript required.

2. The Preamble and Package Options

The minimal preamble for this package is

```
\usepackage[⟨driver⟩,⟨options⟩]{fitr}
```

Permissible names for ⟨*driver*⟩ are `dvips`, `pdftex`, `luatex`, `xetex`, `dvipdfm`, and `dvipdfmx`. The ⟨*driver*⟩ option is only required for `dvipdfm` and `dvipdfmx`, the others are either detected automatically, or is the default driver (`dvips`). The drivers and options are discussed in more detail below.

Package requirements. The hyperref package is brought in through the eforms package. Another package requirement is collectbox by Martin Scharrer. The `\collectbox` command is used to collect the second argument of `\jdRect`, see the discussion of `\jdRect` in Section 3. As a result, the second argument may contain verbatim text in it. Very cool.

[1] The eforms package itself brings in other packages, including hyperref and insdljs.

Package options. The package has several options, broken down below by category: Driver, View, and Other options.

dvips *pdftex* *luatex* *xetex* *dvipdfm* *dvipdfmx*

Driver Options: The major driver options are `dvips` (the default), `pdftex`, `luatex`, and `xetex`. Lesser driver options are `dvipdfm` and `dvipdfmx`, but there seems to be a bug in the `\includegraphics` command as the bounding box is not correctly re-scaled. If you specify `dvips`, it is assumed that you are using Adobe Distiller as your PDF creator.[2]

The fitr package checks whether the web package is loaded, if so, the driver of web is used; otherwise fitr auto-detects for pdftex, luatex, and xetex. If no driver is passed, and neither pdftex, luatex, nor xetex are detected, the default driver, dvips, is used.

preview *viewMagWin*

Viewing Options: When you specify `preview`, the bounding boxes of the buttons are shown in the dvi-previewer (or the PDF viewer); you can turn off this preview by specifying `!preview` (or removing `preview` entirely from the option list). The other option type is `viewMagWin`, when this option the viewing window, a rectangular region, becomes visible in the dvi-previewer (or in the PDF document); specifying `!viewMagWin` turns off this type of preview.

The effects of the viewing options will be illustrated later in this document, see Example 4.1 on page 8.

blinkonjmp *blinkonrestore* *blink*

Special Effects Options: There are several options for bringing in additional document JavaScript for special effects: (1) `blinkonjmp` blinks the border following the jump to a focused destination; (2) `blinkonrestore` blinks the border following the restore action; (3) `blink` is a convenience option that combines the actions of `blinkonjmp` and `blinkonrestore`. There are negative versions of these options as well: `!blinkonjmp`, `!blinkonrestore`, and `!blink`. The default is to have no special effects, in effect `!blink` is the default. Refer to Section 5 for more information on special effects.

gonative

Other Options: The `gonative` option does not use any field or document JavaScript; as a result, the dvips -> ps2pdf is then supported as no document JavaScripts are employed.

3. The `\jdRect` command

The fitr has only one command, `\jdRect`, which (optionally) **j**umps to and/or sets a **d**estination of a Fit**Rect**angle.[3] There are two forms of usage. `\jdRect` optionally creates a push button or link, and optionally creates a viewing window. The term *viewing window* refers to a rectangular region that is created by the **FitR** destination viewing specification, see **Table 8.2 Destination syntax** of the *PDF Reference*, version 1.7. A *named destination* is created and is associated with the viewing window. When we jump to a viewing window, this window is magnified to the largest extent possible.

[2]Unless you are also using the `gonative` option, see the description of this option.
[3]I had some problems naming this command.

For example, click on the either of the two displayed forms of the syntax for `\jdRect`; after jumping to the viewing window, click on the same display to return to the previous view.

There are two versions of `\jdRect`, the command itself, and a * version, `\jdRect*`. The syntax follows, along with the expected parameters.

`\jdRect[⟨key-values⟩]` (Click on box to zoom in)

The above version is used to overlay a region with a button and view window. No content is specified, but is defined by specifying the `width` and `height`; it can be positioned using `shift` and `lift`.

There is a *-version as well:

`\jdRect[⟨key-values⟩]{⟨content⟩}` (Click on box to zoom in)

The second parameter ⟨*content*⟩ is required when the * is present. This version is meant to enclose ⟨*content*⟩ within the button and view window. The `width` and `height` keys are ignored, but `shift` and `lift` are obeyed (though you may `shift` or `lift` the button/view window away from the content).

Description of ⟨*key-values*⟩. Before illustrating the `\jdRect` command, we first discuss its key-value pairs.

- `lift=⟨length⟩`: This key-value lifts (raises) the button/viewing window up (or down); for example, `lift=15pt` (or `lift=-15pt`). The default is a lift of `0pt`. See Example 4.2 on page 8.
- `shift=⟨length⟩`: The amount of horizontal shift; positive to the right, negative to the left. For example, `shift=-1in` shifts the button/viewing window 1 inch to the left. The default is `0pt`. See Example 4.2 on page 8.
- `width=⟨length⟩`: When using `\jdRect`—as opposed to `\jdRect*`—, the width of the button and viewing window is determined by the `width` key. For example `width=1in` creates a button/viewing window that is 1 inch wide. The value of this key is ignored when the * form of the `\jdRect` is used. The default value is `0pt`. This key is required when * is not present. See Example 4.2 on page 8.
- `height=⟨length⟩`: Similar comments here as was made for the `width` key. This key is used to set the height of the button/viewing window. The default is 0pt. It is required when * is not present. See Example 4.2 on page 8.
- `ref=t|c|b`: The `ref` key-value pair determines the reference point of the button/viewing window. Permissible values are `t` top (the default), `c` center, and `b` bottom. This key is only obeyed with the `\jdRect*` form of the command; otherwise, a reference point of `b` is used.
- `adddestw=⟨length⟩`: The default is for the viewing window to have the same dimensions as the underlying button. The `adddestw` key-value pair is used to

widen the viewing window; `adddestw=.2in` widens the window by `.2in` on the left and `.2in` on the right. See Figure 1, page 6.

- `adddesth=`⟨*length*⟩: Similar to `adddestw` but for height. The `adddesth` key-value pair is used to increase the height the viewing window; `adddesth=.2in` increases the height of the window by `.2in` on the top and `.2in` in the bottom. See Figure 1 below.

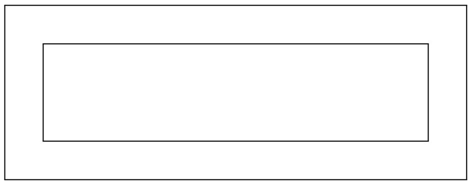

Figure 1: Button and Viewing Window
`width=2in,height=.5in,adddestw=.2in,adddesth=.2in`

- `button=true|false`: `button` is a Boolean switch. If `true` (the default), `\jdRect` creates a push button. When the user pushes the button, the viewer zooms in to the view window. Clicking the same region again restores the previous view.

  When `button` is `false`, the button is not created, but the viewing window is still created. You can then jump to the viewing window with a separate link or button. When `button=false`, use the `dest` key to assign a name (⟨*name*⟩) to the viewing window. (fitr automatically creates the destination names internally, they are used by the buttons. If no button is created, name the destination so you know its name and can reference it in link that jumps to that viewing area.)

- `link=jmp|restore` If `link` has a value, then fitr puts `button=false`. The `link` key is used to create jumps or restore actions to or from a viewing window. When `link=jmp` a jump action is created, the jump will be to the value of the `dest` key. If this is a pure link that jumps to another viewing window, then use the `nodest` key as well; no viewing window will be created around the link, as it is unlikely you'll want to jump to a link.

  For example click on the link **Carl Runge** and jump to the picture of Runge in the margin. Click on the picture of Runge and return to the previous view.

Carl Runge

  The jump to the picture from the text "Carl Runge" is as follows:

```
\jdRect*[nodest,link=jmp,dest=rungePic,
adddestw=10bp,adddesth=10bp]{Carl Runge}
```

  The important options are `nodest,link=jmp,dest=rungePic`; no (named) viewing window is created, we want to create a jump link here, the destination of the jump link is the destination `rungePic`.

The color of the link is determined by `\@linkcolor`, a `hyperref` command that holds a named color. This can be redefined at anytime, directly using

```
\makeatletter
\def\@linkcolor{blue}
\makeatother
```

or, if you are using the `pro` option with the `web` package, you can say,

```
\selectColors{linkColor=blue}
```

When using the `web` package, the default is `webgreen`.

The action to restore the previous view is as follows:

```
\marginpar{\raggedleft\jdRect*[link=restore,dest=rungePic,
  adddestw=\marginparsep,adddesth=\marginparpush
  ]{\parbox[t]{1in}{\RungePic\\
  \normalcolor\centering\footnotesize\textsf{Carl Runge}}}}
```

The picture is placed in the margin using `\marginpar`; the command `\RungePic` is a convenience macro that uses `\includegraphics` in import the picture. The important options are `link=restore,dest=rungePic`, this first key-value pair causes `\jdRect` to create a restore link, the second one says to create a viewing window with a name of `rungePic`, this is the destination the Carl Runge link jumps to.

- `nodest`: A Boolean switch whose default value is `false`. When `nodest` is used (making the switch a value of `true`), no viewing window is created.
- `dest=`⟨*name*⟩: This key is a way of explicitly naming the viewing window (the destination). The destination is normally automatically generated when `button=true`, this key is used with the `link` key, as illustrated above.
- `allowFX`: A Boolean switch (of sorts). The `fitr` allows for special effects (FX) when a viewing window is jumped to and when the view is restored. The default value of `allowFX` is `true` allow special effects if there are any defined. By saying `allowFX=false`, no special effects are used, even if some are defined.

  An example of special effects you say? Try clicking on the Pythagorean Theorem $a^2 + b^2 = c^2$ The default is to allow special effects, the default is set by the command,

  ```
  \renewcommand{\allowFXDefault}{true}
  ```

  By declaring `\renewcommand{\allowFXDefault}{false}`, the default is now set to no special effects.

Passing KV-pairs to the push button of `\jdRect`. When `\jdRect` produces a push button, you can pass `eforms` key-value pairs (KV-pairs) to the underlying push button.

```
\newcommand{\overlayPresets}{\H{I}\BG{}\BC{}\S{S}}
\restoreOverlayPresets
```

See the `eforms` manual for the meaning of these cryptic symbols. The initial value of `\overlayPresets` create transparent push buttons. In subsequent examples, we will redefine `\overlayPresets` to change the visibility of the overlay buttons. The command `\restoreOverlayPresets` returns any re-definition of `\overlayPresets` to its default definition, which is shown above.

4. Some Examples

In this section, several examples are presented that illustrate the options of the `\jdRect` command. The examples in this section are created with

```
\renewcommand{\allowFXDefault}{false}
```

in force. There is one example where we set `allowFX=true`. Additional discussion on special effects (FX) may be found in Section 5.

Example 4.1. Illustrate Preview Rectangles. The `preview` and `viewMagWin` options just set Boolean switches. In this example, we manually give these switches a value of `true` (`\previewtrue\viewMagWintrue`). Take a close look at the following function $f(x) = \dfrac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}\,\mathrm{d}t$, or the more general form $f(x;\mu;\sigma) = \dfrac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\,\mathrm{d}t$ The preview rectangles are shown: For the one on the left, the dimensions of the push button and the viewing rectangle are the same; for one on the right, the dimensions of the viewing window have been increased by using `adddestw=20bp,adddesth=10bp`. When you jump to each of these viewing windows, the one on the left is magnified much more than the one on the right, but the larger viewing window allows the user to see some of the surrounding text. □

Example 4.2. Display Math. Displayed math presents a problem. We take the following set of equations to illustrate.

Suppose we want to classify third order `Runge-Kutta` type methods. Start with

$$\begin{aligned}
K_1 &= hf(t_n, y_n)\\
K_2 &= hf(t_n + rh, y_n + aK_1)\\
K_3 &= hf(t_n + sh, y_n + bK_1 + cK_2)\\
K &= w_1K_1 + w_2K_2 + w_3K_3\\
y_{n+1} &= y_n + K
\end{aligned}$$

Find the system of equations satisfied by $r, s, a, b, c, w_1, w_2, w_3$ that will make the above algorithm a third order method.

The verbatim listing of this set of aligned equations is,

```
1  \begin{align*}
2  \jdRect[height=1.3in,width=2.6in,lift=16pt,shift=-15pt,
3      adddestw=10bp,adddesth=10bp]
4  K_1 &= hf(t_n, y_n)\\
5  K_2 &= hf(t_n +r h, y_n+aK_1)\\
6  K_3 &= hf(t_n +s h, y_n+bK_1+cK_2)\\
7  K &= w_1 K_1+ w_2 K_2+ w_3 K_3\\
8  y_{n+1} &= y_n+K
9  \end{align*}
```

This is an example of `\jdRect` (the non-* version), so there is no second argument. In this case, we create our button dimensions using `height=1.3in,width=2.6in`, line (2). Note the positioning of the `\jdRect` command, the upper-left point of the display. We then use `lift=16pt,shift=-15pt` to move the button around to cover the equations, line (2); finally, we increase the dimensions of the viewing window in line (3) with `adddestw=10bp,adddesth=10bp`. Now, how were the values of these keys determined? By trial and error, while the `preview` and `viewMagWin` options were in effect. Below are the same equations with `\previewtrue` and `\viewMagWintrue`, locally invoked:[4]

$$\begin{aligned}
K_1 &= hf(t_n, y_n)\\
K_2 &= hf(t_n + rh, y_n + aK_1)\\
K_3 &= hf(t_n + sh, y_n + bK_1 + cK_2)\\
K &= w_1K_1 + w_2K_2 + w_3K_3\\
y_{n+1} &= y_n + K
\end{aligned}$$

The preview rectangles do not take up any TeX space, so they overlap parts of the paragraph content. When you zoom in, you'll see part of the part of the word "invoked:", as seen in the upper-left corner, at least according to the viewing window preview. Is it so?

After you've set the position of the rectangles, and after all changes have been made to the underlying content, you don't need the preview modes. □

Example 4.3. Customizing the appearance. The among the properties of the underlying push button is to be visible, but does not print. The background and the border are transparent. The default properties of the button overlay are passed to the push button using `\overlayPresets`, introduced earlier on page 8. You can modify these settings locally, within a group, or globally. In this example, we change the border to red dashed line. We redefine `\overlayPresets` as follows:

```
\renewcommand{\overlayPresets}{\H{I}\BG{}\BC{red}\S{D}}
```

In the re-definition, `\BC{red}` is declared (the `xcolor` package is required here for named colors; otherwise, we would say `\BC{1 0 0}`), `\S{S}` is changed to `\S{D}`, which gives a dashed (D) border as opposed to a solid (S) border. Now to illustrate this. My name is

[4]You can also use the commands `\previewOn`, `\previewOff`, `\viewMagWinOn`, and `\viewMagWinOff` as a convenient way of turning on or off these switches.

D. P. Story!; lets increase the viewing window, shall we? My name is D. P. Story!. Keep in mind that we are overlaying a push button; if you want the underlying text to have a color, you need to color it yourself: D. P. Story! This last button has code,

```
\jdRect*[allowFX,adddestw=10bp,adddesth=10bp]%
    {\textcolor{blue}{D. P. Story}}
```

As the changes to the preset appearance are inside a group, after this example (environment)) `\overlayPresets` will revert to its definition that was in effect outside the example. □

5. Special Effects (FX)

5.1. Built-in FX

There are two built-in JavaScript functions `pbJmpHook()` and `pbRestoreHook()`; the first blinks the view window just after the focused jump, and the latter blinks the view window after the previous view is restored. These built-in functions are not available by default, they are brought into the document with the options `blinkonjmp` and `blinkonrestore`. The option `blink` is equivalent to specifying both `blinkonjmp` and `blinkonrestore`. This document was compiled by inputting the fitr package as `\usepackage[blink]{fitr}`.

• FX examples

Example 5.1. FX and Verbatim. Zoom in on verbatim text `#$$%&$%^&$%^$` which can be handled by `\jdRect*` command. The listing is,

```
\jdRect*[allowFX,adddestw=10bp,adddesth=10bp]
{\verb!#$$%&$%^&$%^$!} which can be handled by
\verb!\jdRect*! command. The listing is,
```

Notice that `allowFX` is specified. This key does not normally appear in the option list, its default value is normally `true`; however, for this document, the following definition was made in the preamble,

```
\renewcommand\allowFXDefault{false}
```

This (re)definition of `\allowFXDefault` sets the default value of `allowFX` to `false`. This is that earlier, the special effects would not be seen, by default. To see their effect, we have to explicitly put `allowFX` to `true`, which is what the single key does. (Or, you can say `allowFX=true`, but that is five more key presses.) □

Example 5.2. FX. Jump to the fitr Package! Click on the verbatim region below to view the listing up close.

```
\renewcommand{\overlayPresets}{\H{I}\BG{}\BC{blue}\S{D}}%
...
Jump to the \jdRect*[allowFX,adddestw=10bp,adddesth=10bp]%
  {{\fitrpkg} Package}
```

We redefined the \overlayPresets command, choosing an initial border of blue, in this way, the document consumer sees that s/he can zoom in on the text.

By the way, you can also zoom in on displayed verbatim text using \jdRect only.

```
\renewcommand{\overlayPresets}{\H{I}\BG{}\BC{blue}\S{D}}%
...
Jump to the \jdRect*[allowFX,adddestw=10bp,adddesth=10bp]%
  {{\fitrpkg} Package}
```

This is done by explicitly enclosing the verbatim region as follows:

```
\jdRect[allowFX,height=4\baselineskip,width=340pt,
  adddestw=10,adddesth=10,lift=-\baselineskip]%
By the way, ...
```

and, in this instance, is placed at the beginning of the sentence. □

5.2. Custom FX

For the standard set up, where there is a push button that overlays the content along with the viewing window it jumps to, there are two JavaScript "hooks" that can be exploited. These are document author defined.

- pbJmpHookCustom() is an undefined JavaScript function that is executed after the jump to the viewing window. (It is enclosed in a try/catch construct that catches the error thrown.) The document author can define pbJmpHookCustom() to perform some action following the jump.
- pbRestoreHookCustom() is an undefined JavaScript function that is executed following the restored view action. The document author needs to make a custom definition if special effects are desired.

The distribution comes with two skeleton custom JavaScript functions, defined in the files jmpHook.js and restoreHook.js. Study the two built-in functions, pbJmpHook() and pbRestoreHook(), which are part of this package, as a guide for writing your own special effects. To use your custom JavaScript functions, place

```
\usepackage[js=jmpHook,js=restoreHook]{lmacs}
```

in the preamble of your document.[5] The lmacs package is a new package I made available to CTAN, it's a simple package that imports files with extensions of .def, .cfg, and .js. We import jmpHook.js and restoreHook.js using a key-value method, where the key is one of the supported extensions; thus js=jmpHook will import the file jmpHook.js if it exists. By the way, another nice feature of lmacs is that you can prefix an exclamation point (!) to cancel out that import, for example, if we wanted to use jmpHook but not restoreHook specify,

```
\usepackage[js=jmpHook,!js=restoreHook]{lmacs}
```

Now, I simply must get back to my retirement. DS

[5]The alternative is to copy and paste the indDLJS environments directly into the preamble.