The AcroSort Package

D. P. Story

1. Introduction

The AcroSort is a novelty LaTeX package for importing a series of sliced images of a picture. The sliced images are randomly arranged, then resorted before the user's eyes using a bubble sort.

Requirements. The techniques used in this package require Acrobat 7.0 Professional, or later. You can probably use pdftex to create the PDF document, but you need Acrobat Pro to import the icons into the document. Since Acrobat Pro is required, you may as well use the Acrobat Distiller!

Press Me Stop Clear

The folder JavaScript file `aeb_pro.js` is used to import the images. This file as special JavaScript coding needed, due to recent (Acrobat 7.0) security restrictions on certain JavaScript methods.

2. Installing `aeb_pro.js`

You need to install `aeb_pro.js`, to do this copy this file to the User's `JavaScripts` folder. To find this folder, execute the script

```
app.getPath("user", "javascript");
```

in the JavaScript Debugger Console window.[1] The return value of this is the path to the `JavaScripts` folder; for example, on my system it returns

```
/C/Documents and Settings/dps
    /Application Data/Adobe/Acrobat/8.0/JavaScripts
```

3. The Method

As you can see from this example, the tiled images are precisely placed on the PDF page, this was done *after* the source files was latexed. After we latex and bring into Acrobat, an `execJS` (from the insDLJS package, loaded by the eForms package) takes over and programmatically, using JavaScript, the icons are imported and the form fields are created and placed precisely were they are supposed to go, with the correct scaling. You must agree, it appears to work well.

Now for the precise placement of the tiled images. You'll note from this document the following lines,

[1] Place the cursor on the line containing this script and press the `Ctrl+Enter` key.

```
\ulCornerHere\reserveSpaceByFile
```

The first one of these creates a form button of `0pt` dimension The button is placed where you want the upper left corner of your image to appear. The JavaScript that is executed when the document is first opened following the distill gets the bounding rectangle of this button and lays out the tiles at that location. Very swave![2]

I sliced the image, `choo.pdf` into 20 tiles (five columns and four rows) in PDF format using the AeB Slicing batch sequence. I also created an `EPS` version of `choo.pdf`. This is the role the command `\reserveSpaceByFile` plays. By including graphic (in draft mode), LaTeX can measure the size of the image by getting its bounding box, and leave room for the tiled images fit together; hence, even though the images are not part of the content of the document, LaTeX can wrap around the image.

4. Package parameters

There are five package parameters, below are the ones used for this document.

```
\theTotalTiles{20}
\theNumRows{4}
\theNumCols{5}
\theImportPath{choo/choo}
\theTeXImageWidth{2in}
```

1. `\theTotalTiles{<number>}`: The total number of tiles (slices) in the image. This command defines a macro `\nTotalTiles` which holds the value of `<number>`.
2. `\theNumRows{<number>}`: The number of rows in the image. This command defines a macro `\nRows` which holds the value of `<number>`.
3. `\theNumCols{<number>}`: The number of columns in the image. This command defines a macro `\nCols` which holds the value of `<number>`.
4. `\theImportPath{<path>}` The path to the file, the end of the path is the base name, without extension of the tiles. The tiles are named as follows: `basename_01`, `basename_02`, ...`basename_10`, `basename_11`, ... This is the naming convention used by the AeB Slicing batch sequence. This command defines a macro `\importpath` which holds the value of `<path>`.
5. `\theTeXImageWidth{<length>}` The image is not necessarily displayed using its natural dimensions. This parameter allows you to specify its rescaled width. This command defines a macro `\texImageWidth` which holds the value of `<length>`.

[2]You can also create your form fields using the eForms package, laying them out in proper rows and columns, you would still need `execJS` to convert them to buttons that have an icon appearance. But this is not half so much fun, and not nearly as swave.

5. After assembly

After you latex the document and bring the newly created PDF into Acrobat, some JavaScript will be executed to import the icons and to lay out the button form fields. After things have settled down, be sure to *save the document* before distribution.

Use the PDF Optimizer, under the Advance menu, to reduce the size of the file. For example, the file size of this document after distillation was 1.89 MB, but after running the PDF Optimizer on it, the file size was reduced to 176.7 KB!

6. Limitations

This package uses much of the same code as the AcroMemory package. These two packages cannot be used together for one document, and it makes no sense to use them together.

Only one image can be randomized and sorted per document. The package can be generalized to have more than one, but the novelty would wear a little thin, in this case.

7. Uses

i’ve used this package to create birthday greetings cards for friends. You can use it for whatever novel idea your mind can conjure up! Enjoy!

Now, I simply must get back to my retirement, dps