AcroTEX.Net

The acrosort Package

D. P. Story

Distribution Dated: 2020/06/17
www.acrotex.net
Version 1.6.1

Table of Contents

1. Introduction

tiled bubble sort

acrosort is a novelty LaTeX package for importing a series of tiled images of a picture. The tiled images are randomly arranged, then resorted before the user's eyes using a bubble sort. We shall refer to the figure to the left as a *tiled bubble sort*.

This new version of acrosort, dated 2020/06/02 or later, supports all common workflows: pdflatex, lualatex, xelatex, and dvips -> distiller.

Press Me Clear

multiple tiled bubble sorts supported

The graphicx, eforms, and icon-appr packages are automatically input by acrosort. When the workflow dvips -> distiller is used, the package aeb_pro is required. For the first time, multiple tiled bubble sorts can appear in the same document, though only one can be sorted at a time.

Demo files. There are two sample files for this distribution: `as1.tex` (only one tile bubble sort), and `as2.tex` (two tiled bubble sorts). These are found in the `examples` folder.

2. The Method

The creation of the *tiled bubble sort* has two easy steps. `:-{)`

1. Embed your graphics using the `embedding` environment of icon-appr and the special command `\asEmbedTiles`.

```
\begin{embedding}
\isPackage
\asEmbedTiles[⟨ext⟩]{⟨name⟩}{⟨num-tiles⟩}{⟨path⟩}
\end{embedding}
```

The `\asEmbedTiles` is defined by acrosort. The parameters are: ⟨*name*⟩ is a unique name used by `\insertTiles` to refer this tile embedding; ⟨*num-tiles*⟩ is the number of tiles; ⟨*path*⟩ is the path to the graphics file,[1] the graphics file is referenced by its *base name*. Usually, PDF files are used for graphics.

`\isPackage` is optional and must appear prior to the `\asEmbedTiles` command to which it refers. `\isPackage` means the tiled graphic files are "packaged" in a single PDF, named ⟨*base name*⟩`_package.pdf`.

The optional argument ⟨*ext*⟩ is ignored when `\isPackage` is present; otherwise, an extension of ⟨*ext*⟩ is affixed to the graphics file. If ⟨*ext*⟩ is not specified, then an extension of `pdf` (`.pdf`) is assumed.

Base name: Suppose the base name is `myPicure`, then ⟨*path*⟩ might be `graphics/myPicture`. If `\isPackage` is expanded prior to `\asEmbedTiles`, acrosort looks

---

[1] a relative or absolute path, relative preferred

for `myPicture_package.pdf` in the `graphics` folder. If `\isPackage` does not appear, then acrosort looks for the sequence of the tiled graphic files `myPicture_01`, `myPicture_02`, …`myPicture_`⟨*num-tiles*⟩, numbers less than 10 are prefixed with a zero (0). In this case, the graphic file extension is taken to be the one specified by ⟨*ext*⟩, or as `.pdf`, otherwise.

`\asEmbedTiles` puts the base graphic file in a box and measures its dimensions; the format for the base graphic must be in a format that `\includegraphics` supports, for whatever PDF creator you are using. In particular, when using straight LaTeX, the base document show have an EPS version. Note that in each of the graphics folders (`choo` and `emoji`) both PDF and EPS versions of the base graphic are provided.

2. In the body of the document, place the `\insertTiles` command:

```
\insertTiles{⟨name⟩}{⟨width⟩}{⟨n-rows⟩}{⟨n-cols⟩}
```

where ⟨*name*⟩ is the name of an embedding (`\asEmbedTiles`); ⟨*width*⟩ is the total width of the picture; ⟨*n-row*⟩ is the number of rows; ⟨*n-cols*⟩ is the number of columns.

For the tiled bubble sort figure of this document, the following was used.

```
...
\begin{embedding}
\isPackage
\asEmbedTiles{choo}{20}{../examples/choo/choo}
\end{embedding}
...
\begin{document}
...
\insertTiles{choo}{2in}{4}{5}
...
\end{document}
```

It's just that simple !

3. Controlling the bubble sort

Below are three basic commands for controlling a tile bubble sort by the name of ⟨*name*⟩.

```
\StartSort[⟨KV-pairs⟩]{⟨name⟩}{⟨wd⟩}{⟨ht⟩}
\StopSort[⟨KV-pairs⟩]{⟨wd⟩}{⟨ht⟩}
\ClearSort[⟨KV-pairs⟩]{⟨name⟩}{⟨wd⟩}{⟨ht⟩}
```

Use ⟨*KV-pairs*⟩ to change the appearance of the fields, where ⟨*KV-pairs*⟩ are eforms field key-value pairs. The ⟨*name*⟩ argument (`\StartSort` and `\ClearSort`) is the name of the graphics to be controlled. (⟨*name*⟩ must match up with the ⟨*name*⟩ argument of

`\asEmbedTiles` and `\insertTiles`.) The ⟨*wd*⟩ and ⟨*ht*⟩ are the width and height of the push button fields. If a caption is provided, set ⟨*wd*⟩ to `{}` and `eforms` will automatically calculate the width based on the value of the `\CA` key.

There are several other commands of interest, these are,

`\customStartJS{⟨script⟩}` (Field level) Inserts ⟨*script*⟩ just prior to the start of the sort (`\StartSort`). The default is `\customStartJS{}`.

`\customFinishJS{⟨script⟩}` (Document level) Inserts ⟨*script*⟩ just after the finish of the sort. The default is `\customFinishJS{}`.

`\appendStartSortJS{⟨script⟩}` (Field level) Inserts ⟨*script*⟩ following the underlying package JavaScript of `\StartSort`. The default is `\appendStartSortJS{}`.

`\appendStopSortJS{⟨script⟩}` (Field level) Inserts ⟨*script*⟩ following the JavaScript of `\StopSort`. The default is `\appendStopSortJS{}`.

`\appendClearSortJS{⟨script⟩}` (Field level) Inserts ⟨*script*⟩ following the underlying package JavaScript of `\ClearSort`. The default is `\appendClearSortJS{}`.

Some simple examples; assume there is a text field by the name of `"message"`:

```
\renewcommand{\customStartJS}{%
    var f=this.getField("message");
    f.value="Begin sorting choo";
}
\renewcommand{\customFinishJS}{%
  var f=this.getField("message");
  f.value="Finished sorting choo";
}
```

Placement. It should be noted that the above commands marked as “Field level” may be placed in the body of the document, prior to the commands they effect. The other command (`\customFinishJS`), which is marked as “Document level,” needs to be placed in the preamble to have any effect.

The sample file `as2.tex` provides examples of these various commands.

4. Creation of tiles

tile-graphic pkg

Use the package `tile-graphic` to tile a graphics file. In the `examples` folder there are two demo files, `as1.tex` and `as2.tex`, that use the graphics in the `emoji` and `choo` folders. These two folders contain files `tg-emoji.tex` and `tg-choo.tex` that were used to produce the tiled graphics. The one in the `emoji` folder is reproduced below.

`pdfcreator=`
`pdflatex|`
`lualatex|`
`xelatex|`
`distiller`

```
\documentclass{article}
\usepackage[!wrttofiles,packagefiles,pdfcreator=pdflatex]{tile-graphic}
\setTileParams{4}{4}{emoji}
\begin{document}
\tileTheGraphic
\end{document}
```

for xelatex users

Refer to the documentation of tile-graphic for more information. Currently, the options are `!wrttofiles` and `packagefiles`, these produce `emoji_package.pdf`. If you are using `xelatex`, you'll need the "non-packaged" files. Produce them by changing the options to `wrttofiles` and `!packagefiles` and compile, the files `emoji_01.pdf`, `emoji_02.pdf`, …, `emoji_12.pdf` should be created. It's just that simple!

5. Applications

I've used this package to create birthday, wedding, and anniversary cards for friends. You can use it for whatever novel idea your mind can conjure up! Enjoy!

Press Me Clear