The animate Package

Alexander Grahn
a.grahn@web.de

4th May 2007

Abstract

A LaTeX package for creating portable, JavaScript driven PDF animations from sets of graphics files or inline graphics.

Keywords: include, portable, animation, animating, embed, animated, inline graphics, vector graphics, graphics files, LaTeX, dvips, ps2pdf, pdfLaTeX, PDF, JavaScript, PSTricks, pgf, TikZ, LaTeX-picture, Adobe Reader

Contents

1 Introduction

This package provides an interface to create PDFs with animated content from sets of graphics files, from inline graphics, such as LaTeX-picture, PSTricks or pgf/TikZ generated pictures, or just from typeset text. Unlike standard movie/video formats, which can be embedded, for example, using the LaTeX package 'movie15' [1], package 'animate' allows for animating vector graphics. The result is roughly similar to the SWF (Flash) format, although not as space-efficient.

Package 'animate' supports the usual PDF making workflows, i. e. pdfLaTeX and LaTeX → `dvips` → `ps2pdf` (Ghostscript).

The final PDF can be viewed in current Adobe Reader®s on all supported platforms.

The 'animate' package makes use of Optional Content Groups (OCG), also known as PDF layers, which is a feature provided by the PDF-1.5 specification. Each frame of an animation is associated with an OCG that is made visible or invisible in a dynamic fashion by means of Adobe Reader's built-in JavaScript engine.

2 Requirements

pdfεTeX, version ≥ 1.20 for PDF output

Adobe Reader, version ≥ 6

3 Installation

The file 'animate.sty' should be stored in a place where LaTeX can find it.

4 Using the package

First of all, read Section 7 on problems related to this package. Then, invoke the package by putting the line

```
\usepackage[<package options>]{animate}
```

to the preamble of your document, i. e. somewhere between `\documentclass` and `\begin{document}`.

'animate' honours the package options:

```
autoplay
autopause
autoresume
controls
buttonsize=<size>
```

```
loop
palindrome
screenbg=<colour>
buttonbg=<colour>
step
poster=[first | none | last]
```

The same options are also available as command options and will be explained shortly. However, if used as package options they have global scope, taking effect on all animations in the document.

If PDF is generated via DVI and Postscript by the command sequence `latex` → `dvips` → `ps2pdf`, the 'graphicx' package is required. *Important:* The `dvips` option '`-Ppdf`' should *not* be set when converting the intermediate DVI into Postscript. If you cannot do without, put '`-X 2400 -Y 2400`' *after* '`-Ppdf`' on the command line.

5 The user interface

Package 'animate' provides the command

```
\animategraphics[<options>]{<frame rate>}{<file basename>}{<first>}{<last>}
```

and the environment

```
\begin{animateinline}[<options>]{<frame rate>}
    ... typeset material ...
\newframe
    ... typeset material ...
\newframe*
    ... typeset material ...
\end{animateinline}
```

While `\animategraphics` can be used to assemble animations from sets of existing graphics files, the environment '`animateinline`' is intended to create the animation from the typeset material it encloses. This material can be pictures drawn within the LaTeX `picture` environment or using the advanced capabilities of PSTricks or pgf/TikZ. Even ordinary textual material may be animated in this way. The command `\newframe` terminates a frame and starts the next one. There is a starred version, `\newframe*`. If placed after a particular frame it causes the animation to pause at that frame. The animation continues normally after clicking it again.

The parameter `<frame rate>` specifies the number of frames per second of the animation. All files of the sequence must be consecutively numbered. `<file basename>` is the leftmost part of the file name that all members of the sequence have in common. `<first>` is the number of the first and `<last>` the number of the last file in the set. `<first>` and `<last>` must have the same number of digits to ensure proper sorting. If necessary, rename the files with zeros padded to the left.

There is no file name extension to be specified as parameter. The possible file formats depend on whether LaTeX or pdfLaTeX is used. In the case of LaTeX, files with the extension '`eps`' are searched for at first, followed by '`mps`' (METAPOST-generated Postscript) and '`ps`'. With pdfLaTeX the searching order is: (1) '`pdf`', (2) '`mps`', (3) '`png`', (4) '`jpg`', (5) '`jpeg`', (6) '`jbig2`' and (7) '`jb2`'. That is, files capable to store vector graphics are found first. Make sure that all file names have *lower case* extensions.

For example, given the sequence 'frame_05.png' through 'frame_19.png' from a possibly larger set that shall be used to build the animation. Then, `<file basename>` would be specified as '`frame_`', `<first>` as '`05`' and `<last>` as '`19`'.

The following options to `\animategraphics` and '`animateinline`' have been provided:

`autopause`

Pause animation when the page is closed, instead of stopping and rewinding it to the default frame.

`autoplay`

Start animation after the page has opened. Also resumes playback of a previously paused animation.

`autoresume`

Resume previously paused animation when the page is opened again.

`loop`

The animation restarts immediately after reaching the end.

`palindrome`

The animation continuously plays forward and backward.

`controls`

Inserts control buttons below the animation widget. The meaning of the buttons is as follows, from left to right: stop & first frame, step backward, play backward, play forward, step forward, stop & last frame, decrease speed, default speed, increase speed. Both 'play' buttons are replaced by a large 'pause' button while the animation is playing.

`width=<width>`
`height=<height>`
`depth=<depth>`

Resize the animation widget. Any valid TeX dimension is accepted as parameter. Option '`depth`' specifies how far the animation widget should extend below the bottom line of the running text. If only one of these options is given, the others are scaled to keep the aspect ratio.

`scale=<factor>`

Scales the animation widget by `<factor>`.

`buttonsize=<size>`

Changes the control button height to `<size>`, which must be a valid TEX dimension. The default button height is 1.44em and thus scales with the current font size.

`screenbg=<colour>`
`buttonbg=<colour>`

The background of the animation and control button widgets is transparent by default. It can be turned into a solid colour by the parameter `<colour>`, which is an array of numbers (*without* surrounding brackets) in the range from 0.0 to 1.0. The number of array elements determines the colour model in which the colour is defined: (1) gray value, (3) RGB, (4) CMYK.

`step`

Step through the animation by one frame per mouse-click.

`poster=[first | none | last]`

Specifies which frame (first, last or none) to display and print if the animation is not activated. Option 'poster=first' need not be explicitly set because it is the default.

6 Examples

6.1 Animation from a set of files, using 'animategraphics' command

```
\documentclass{article}
\usepackage{animate}
\usepackage{graphics}

\begin{document}

\begin{center}
  \animategraphics[controls,loop]{4}{frame_}{0}{8}
\end{center}

\end{document}
```

6.2 Animating PSTricks graphics, using 'animateinline' environment

```
\documentclass{article}
\usepackage{pst-3dplot}
\usepackage{fp}
\usepackage{animate}

%draws a torus sector
\newcommand{\torus}[1]{% #1: angle of the torus sector
  \psset{Beta=20,Alpha=50,linewidth=0.1pt,origin={0,0,0},unit=0.35}%
  \begin{pspicture}(-12.3,-6.3)(12.3,7)%
    \parametricplotThreeD[xPlotpoints=100](80,#1)(0,360){%
      t cos 2 mul 4 u sin 2 mul add mul
      t sin 2 mul 4 u sin 2 mul add mul
      u cos 4 mul
    }%
    \parametricplotThreeD[yPlotpoints=75](0,360)(80,#1){%
      u cos 2 mul 4 t sin 2 mul add mul
      u sin 2 mul 4 t sin 2 mul add mul
      t cos 4 mul
    }%
    \FPupn\strokewidth{360 #1 sub 360 div 3 mul}%
    \parametricplotThreeD[yPlotpoints=1,linewidth=\strokewidth pt](0,360)(#1,#1){%
      u cos 2 mul 4 t sin 2 mul add mul
      u sin 2 mul 4 t sin 2 mul add mul
      t cos 4 mul
    }%
  \end{pspicture}%
}

\begin{document}

\begin{center}
\newcounter{torusangle}
\setcounter{torusangle}{80}
```

```
\begin{animateinline}[poster=last,controls,palindrome]{10}%
  \torus{\thetorusangle}%
  \whiledo{\thetorusangle<360}{%
    \newframe%
    \addtocounter{torusangle}{10}%
    \torus{\thetorusangle}%
  }
\end{animateinline}%

\end{center}

\end{document}
```

7 Bugs

- The command `\multido` and its relatives from package '`multido`' do not work if the loop body contains `\newframe`. Use `\whiledo` from package '`ifthen`' instead. A counter must be declared to keep track of the number of iterations. Further variables that take fixed point decimals to be used within the loop body can be defined and incremented by means of the commands `\FPset` and `\FPadd` from package '`fp`'. For example:

```
% declare loop counter
\newcounter{iter}%

\begin{animateinline}{12}
  % initialize loop counter
  \setcounter{iter}{0}%
  % define some variable, e. g. `\somevar'
  \FPset{\somevar}{1.0}%
  %make copy `\varcopy' of `\somevar' with global scope
  \xdef\varcopy{\somevar}%
  ...
  ... create first frame using \somevar(=1.0)
  ... (\multido _can_ be used here!)
  ...
  \whiledo{\theiter<50}{% do 50 iterations
```

```
    % start new frame
    \newframe%
    %increment loop counter by 1
    \addtocounter{iter}{1}%
    %increment \somevar by, e. g., 0.5
    \FPadd{\somevar}{\varcopy}{0.5}%
    %refresh the copy
    \xdef\varcopy{\somevar}%
    ...
    ... create frame using \somevar
    ... (\multido _can_ be used here!)
    ...
  }
\end{animateinline}
```

- Currently, animations cannot be placed on multilayered slides created with the Beamer class. Put animations on flat slides only. (Of course, slides without animations may still have overlays.)
- The `dvips` option '`-Ppdf`' should be avoided entirely or followed by '`-X 2400 -Y 2400`' on the command line in order to set a sensible DVI resolution. In times of Type-1 fonts, this does *not* degrade the output quality! The configuration file '`config.pdf`' loaded by option '`-Ppdf`' specifies an excessively high DVI resolution that will be passed on to the final PDF. Eventually, Adobe Reader gets confused and will not display the frames within the animation widget.
- Animations with complex graphics and/or many frames may cause LaTeX to fail with a '`TeX capacity exceeded`' error. Enlarge TeX's memory with command line option '`--mem-max=...`'.
- Originally, package '`animate`' was based on a method suggested by Jan Holeček and Petr Soika [2], but was rewritten to use Optional Content Groups (OCG). OCG-based animations are fully supported in Adobe Reader 8, but have some performance issues. Playback is usually slower in Reader 8 than in older Reader versions, which is probably a bug. The non-OCG approach performs better in Adobe Reader 8, but package/command options '`autoplay`', '`autopause`' and '`autoresume`' do not work. Moreover, the animation position and the play/pause' buttons are not properly reset on page change. This is due to usage restrictions that Adobe imposed on some important JavaScript methods in Reader 8. A non-OCG version of the package is still available as '`animate-noocg.sty`'.

8 Acknowledgements

I would like to thank François Lafont who discovered quite a few bugs and made many suggestions that helped to improve the functionality of the package.

References

[1] *The Movie15 Package*. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/movie15`

[2] Holeček, J. ; Sojka, P.: Animations in pdfTEX-generated PDF. In: *Lecture Notes in Computer Science*, Vol. 3130, 2004, pp. 179–191