

The movie15 Package

Alexander Grahn
a.grahn@web.de

16th May 2012

Abstract

A L^AT_EX package for inclusion of movies, sounds and 3D objects into PDF documents with PDF-1.5/1.6 compatibility.

Keywords: embed, movie, LaTeX, pdfLaTeX, PDF, 3D, JavaScript, include, sound, video, multimedia, animation

This package is obsolete now. Consider using package ‘media9’.

Contents

1	Introduction	1
2	Installation	2
3	Requirements	2
4	Using the package	2
5	The user interface	3
5.1	Media inclusion	3
5.2	Inclusion of 3D objects	7
5.3	Media hyperlinks	12
5.4	Compatibility commands	14
6	Examples	15
A	3D quick-start guide	16
B	Media formats	19

1 Introduction

The way multimedia content, i.e. movies and sounds, is included into PDF has changed with Adobe’s PDF specification, version 1.5. PDF-1.5 supports a

larger variety of movie and sound formats, limited only by the number of plugins available for Adobe Reader[®]. With PDF-1.6, support for 3D objects, stored in the U3D file format, has been added. See Table 3 for a list of viable media formats. The specification allows media file contents to be completely embedded into the PDF output, thus producing self-contained PDF documents.

This package provides an interface to embed movies, sounds and 3D objects into PDF documents for use with L^AT_EX as well as pdfL^AT_EX. Media file contents is incorporated into PDF output by default. This is done either directly during source processing using pdfL^AT_EX or during conversion from Postscript to PDF using Ghostscript's `ps2pdf`.

The final PDF output can be viewed with Windows[®] and Mac OS[®] versions of Adobe Reader. However, embedded media file data can also be extracted and saved to disk from within Readers which support file attachments. This makes PDF documents a little more portable to Readers which do not ship with a multimedia plug-in.

'movie15' works well together with the presentation making package 'Beamer'. In particular it supports its overlay concept. See this example. Also read the comments in the example's source file, '`overlay-example.tex`', on how to use 'movie15' with 'Beamer'.

2 Installation

The file '`movie15.sty`' should be stored in a place where L^AT_EX can find it.

3 Requirements

pdfT_EX, version ≥ 1.20 , is needed for direct PDF output.

If the package option '3D' is set, 'movie15' loads Michael Mehlich's 'fp' package for fixed point arithmetic. Since it does not belong to the core packages of most T_EX distributions it must be installed prior to selecting the '3D' option.

Adobe Reader, version 6, is required for playing movies and sounds, version 7 for rendering embedded 3D content.

4 Using the package

Invoke the package by putting the line

```
\usepackage[<package options>]{movie15}
```

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

'movie15' honours the following package options:

3D

Enables the 3D feature from the PDF-1.6 specification. Inclusion of 3D files into PDF is discussed separately in section 5.2.

`draft`

Media files are not included. Just the file name is printed in a box of size $\langle\text{width}\rangle\times(\langle\text{height}\rangle+\langle\text{depth}\rangle)$.

`final`

The opposite of `draft`. Useful to override a global `draft` option specified in the `\documentclass` command.

If PDF is generated via DVI and Postscript by the usual `latex` \rightarrow `dvips` \rightarrow `ps2pdf` sequence of commands, the ‘`graphicx`’ package is required.

Note that several runs may be necessary to resolve internally created cross-references. Appropriate warnings will be issued in such cases.

5 The user interface

5.1 Media inclusion

Movies, sounds and 3D objects are embedded into the document using the command

```
\includemovie[<options>]{<width>}{<height>}{<media file>}
```

Unless left empty, the `<width>` and `<height>` arguments must be given in valid $\text{T}_{\text{E}}\text{X}$ dimensions. Horizontal and vertical dimensions of the media clip are scaled independently to fit `<width>` and `<height>` or, if the two latter are empty, to fit the size of the text box, given as argument to the ‘`text`’ option; see below. `<media file>` specifies the file name of the media clip. If the media file is embedded as part of the final PDF output, which is the default, it may reside wherever $\text{T}_{\text{E}}\text{X}$ or Ghostscript search for input files, depending on the PDF producing method.

Below, common options to `\includemovie` are listed. Options specific to embedding of 3D content are discussed separately in section 5.2. The impatient reader may advance directly to the 3D quick-start guide, which summarizes the basic steps for embedding 3D objects.

`attach[=false]`

By default, the embedded data stream is re-used as embedded file attachment. This makes ‘`movie15`’ a little more portable to PDF Readers which do not have a multimedia plug-in because it gives the user the opportunity to extract the media file and to save it to disk for later playback with an external player. However, support of file attachments is required. Re-use as file attachment can be suppressed with option ‘`attach=false`’. Note that ‘`attach=false`’ is a poor method to prevent the user from obtaining a copy of the media file. There are

several (free) tools for decompressing and extracting stream objects from PDF files.

autoclose

Close the media player when the page is closed. Use of this option may be necessary for streamed media, such as Real Video.

autopause

Pause playing when the page is closed.

autoplay

Start playback of the media clip after the page has opened. This is especially useful if there are multiple movie clips on the current page to be played at the same time. Also resumes playback of previously paused media. In the case of embedded sound being started this way, `<width>` and `<height>` can safely be set to '0pt' since no further user interaction is necessary to start playing.

autostop

Stops playing but keeps the media loaded when the page is closed. The media is rewound to its beginning or its `'startat'` position. This option is set by default if neither of the options `'autopause'`, `'autoclose'` or `'continue'` is set.

autoresume

Resume previously paused playback when the page is opened again.

continue

Continue playback in background on premature page change. However, paused media remain paused. So do media that have reached the end or that still show the poster.

controls

toolbar

Player specific controls or toolbars are displayed during playback of the media clip, if available. The default is `'false'`; controls are not shown.

depth=<depth>

Specifies how far the media display area should extend below the bottom line of the running text. Any valid TeX dimension is accepted.

draft

Locally switches to draft mode.

externalviewer

Opens and plays the media in an external application.

final

Locally disables draft mode.

inline[=false]

Media file *data* are embedded into PDF output by default. This can be suppressed with option ‘`inline=false`’.

`label=<label spec>`

The media is labelled with `<label spec>` such that it can be referenced by the `\movieref` command elsewhere in the document. The `\movieref` command creates a hyperlink to either stop, pause, resume the media, to play it with different settings, such as frame rate or volume, or to show a different part of it. See section 5.3 for details. `<label spec>` may consist of any sequence of letters, digits or punctuation characters.

`mimetype=<mimetype>`

According to the PDF specification, the viewer application needs to know the MIME type of `<media file>`. Package ‘`movie15`’ tries to make a guess from the file name extension. If, however, a media file is of unknown type or has a non-standard file name extension, its MIME type must be explicitly specified using this option. See Table 3 for a list of known file name extensions.

`mouse[=true|false]`

Enable mouse interaction: Clicking onto the media (giving it the focus) pauses the playback while clicking outside resumes it. Alternatively, once the media has got the focus, repeated pressing of key ‘P’ on the keyboard switches between Play and Pause (at least with some players). In 3D context it specifies whether the 3D annotation should be interactive, i. e. responsive to mouse interaction. 3D annotations are interactive by default.

`palindrome[=true|false]`

The media is played forward and backward, if supported by the media player. In combination with option ‘`repeat`’ this forward-and-reverse playback repeats as many times as specified. Each complete forward and reverse playback counts as one repeat.

`playerid=<playerid>`

Forces a particular media player plug-in to be used instead of the default one. Currently, the following values are recognized:

- AAPL_QuickTime
- MACR_FlashPlayer
- RNWK_RealPlayer
- MSFT_WindowsMediaPlayer
- ADBE_MCI (Adobe builtin player)

`poster`

The first frame of the movie or the frame at the beginning of the movie section specified by the ‘`startat`’ option is shown.

`poster=<image>`

Inserts still image from file `<image>` to be shown when the media clip is not activated. The file type of `<image>` depends on the output driver: PS or EPS when using \LaTeX together with `dvips`; PDF, PNG or JPEG when using `pdf \LaTeX` .

This option is obsolescent and should not be used any longer. Instead, use option ‘text’ if a still image from an external file is to be inserted. See option ‘text’ for a possibility to scale the media display area according to the poster image dimensions.

`rate=<rate>`

<rate> specifies the playback rate, where 1 is normal playback, 0.5 is half speed, 2 is double speed, -1 is normal speed in reverse etc.

`repeat [=<repeats>]`

Specifies how often the media clip or embedded 3D animation will be played. Without argument this option causes the player to repeat forever.

`startat=<offset>`

`endat=<offset>`

Start/stop playback at the specified offsets. <offset> may be an absolute time or frame value, or a named marker, or a marker plus a time or frame. In the latter case, time or frame values are regarded as relative to the named marker. Support of markers, time and frame offsets is media type dependent. Media incompatible offsets are ignored. The following syntax applies to <offset>:

```
time:<time in seconds>
frame:<frame number>
marker:<quoted string>
```

Some <offset> examples:

```
time:20.5
frame:100
{marker:'Chapter 1', time:60}
```

`text=<text>`

<text> is typeset onto and centred within the media display area which has been specified by the <width> and <height> arguments as well as the ‘depth’ option. If either or both of the <width> and <height> arguments are left empty, any unspecified dimension of the media display is scaled to match the corresponding dimension, i.e. width and height, of <text>. Likewise, the depth of the media display is adjusted to the depth of <text> if the optional ‘depth’ has not been given. Therefore, option ‘text’ can be used to scale the media display to the natural or scaled dimensions of a poster image file, such as (requires package ‘graphicx’; *also note the empty <width> and <height> arguments!*):

```
\includemovie[text={\includegraphics[scale=2]{path/to/poster}}]
{}{}{path/to/movie}
```

In combination with the ‘poster’ option, which causes the poster image to be rendered from the movie during runtime, the PDF file size can be reduced somewhat by putting `\includegraphics` into a `\phantom` box:

```
\includemovie[
poster,
text={\phantom{\includegraphics[scale=2]{path/to/poster}}}
]{}{}{path/to/movie}
```

`textoverposter[=false]`

Set this option to 'false' if you wish `<text>` to be obscured by the external poster image which has been inserted with the poster option. Deprecated.

`url`

Treat `<media file>` as URL. Sets option 'inline' to 'false' automatically.

`volume=<percentage of original volume>`

The meaning of this option should be self explaining.

5.2 Inclusion of 3D objects

The PDF-1.6 specification, which was introduced with the advent of Adobe Acrobat/Reader 7, allows embedding of 3-dimensional graphic objects, such as CAD models or 3D scientific data, and lets the user interactively manipulate them. At the time of writing this documentation, the only supported file types were U3D [2] and Adobe's PRC format, and only one commercial software [3] for exporting into the U3D format, yet from a number of CAD and 3D vector formats, including DXF and VRML, was known. Nevertheless, a try-out version of [3] can be downloaded without charge.

Selection of the '3D' package option enables the 3D feature. Most of the command options listed in section 5.1 do what they are supposed to do in the case of embedded 3D as well. Other options are ignored, in particular 'startat', 'endat', 'volume' and 'playerid'. Options 'inline=false' and 'url' are supported, but imply option 'externalviewer', because the 3D Reader plug-in handles embedded files only.

There are a few options to `\includemovie` which define how the 3D object is positioned within the view port of a virtual camera, or conversely, how the virtual camera is positioned and oriented within a coordinate system, called 'The World', which bears the 3D object at a fixed position. Fig. 1 should help to visualize the scenery: The virtual camera is orbiting at a distance of ROO around the centre of orbit, specified by the position vector \overrightarrow{COO} ; $\angle AAC$ is the camera's aperture angle. In addition, the direction vector $\overrightarrow{C2C}$ is needed to specify the initial camera position.

The *default view*, i. e. the view that is shown initially after activating the 3D object in the Reader, can be set using the options '3Dcoo' for the centre of orbit, '3Dc2c' for the centre of orbit to camera vector, '3Droo' for the orbital radius and '3Daac' for the aperture angle of the camera:

`3Daac=<angle>`

This option sets the aperture angle of the camera, measured in degrees. Fixed point real numbers between 0 and 180 are admissible. A sensible value of 30 is pre-set by default. Larger values can be used to achieve wide-angle or fish-eye effects. See example 3 in section 6.

`3Dc2c=<x> <y> <z>`

if `<roll>` is greater than zero); measured in degrees.

`3Droo=<r>`

`<r>` (always positive!) specifies the radius of orbit *ROO* of the virtual camera. See option `'3Dc2c'` for the number format of `<r>`.

Without the above options the virtual camera sits at the origin (0,0,0) of the World, looking in the positive *Y* direction, i.e. default settings of `3Droo=0`, `3Dcoo=0 0 0` and `3Dc2c=0 -1 0` are assumed. (Note that $\overrightarrow{C2C}$ is the opposite of the view vector!) Thus, in order to get a 'front view' of the 3D object it is sufficient to set the radius of orbit, i.e. the distance between camera and object appropriately. Sometimes you may want to adjust the orbital centre, i.e. the target of the camera as well, in particular, if the object is irregularly shaped. Fortunately, it is possible to let the values of the corresponding options be determined automatically. Simply insert a hyperlink using the `\movieref` command together with the `'3Dcalculate'` option. Once the settings for `'3Droo'` and `'3Dcoo'` have been calculated, the hyperlink can be removed again. See section 5.3 for an explanation and example 3 in section 6.

`3Dbg=<r> <g> `

This option sets the background colour of the canvas. Only fixed point real numbers in the range from 0 to 1 are allowed for the colour components.

`3Dlights=<lighting scheme>`

Sets the default lighting scheme. The following values are honoured: `'None'`, `'White'`, `'Day'`, `'Night'`, `'Hard'`, `'Primary'`, `'Blue'`, `'Red'`, `'Cube'`, `'CAD'`, `'HeadLamp'`. The default is to use the lighting scheme as specified within the 3D artwork.

`3Drender=<render mode>`

Sets the default render mode. The following values are honoured: `'Solid'`, `'SolidWireframe'`, `'Transparent'`, `'TransparentWireframe'`, `'BoundingBox'`, `'TransparentBoundingBox'`, `'TransparentBoundingBoxOutline'`, `'Wireframe'`, `'ShadedWireframe'`, `'HiddenWireframe'`, `'Vertices'`, `'ShadedVertices'`, `'SolidOutline'`, `'Illustration'`, `'ShadedIllustration'`.

`3Dviews=<views file>`

Deprecated. Superseded by the `'3Dviews2'` option and a new, more flexible views file syntax (see below). File `<views file>` specifies predefined camera positions. It contains lines with the following syntax:

```
[<name>]{<coo_x> <coo_y> <coo_z>}{<c2c_x> <c2c_y> <c2c_z>}{<roo>}{<roll>}{<aac>}
```

The `<name>` entry is optional. If `<name>` is not given, a default name consisting of 'View' followed by the number of the current entry in the list is formed. For `<coo_x>`, `<coo_y>`, `<coo_z>`, `<c2c_x>`, `<c2c_y>`, `<c2c_z>`, `<roo>`, `<roll>` and `<aac>` the same rules as for the corresponding options `'3Dcoo'`, `'3Dc2c'`, `'3Droo'`, `'3Droll'` and `'3Daac'` apply. Empty braces, {}, are possible and cause default values to be used. Trailing spaces or comment signs (%) are allowed. Reading of the file stops either at its end, at the first empty line encountered or at the first line containing nothing but spaces and/or a comment sign followed by arbitrary stuff.

3Dviews2=<views file>

Instead of or in addition to the default view, further *named views* can be set in an auxiliary file ‘<views file>’. Besides the virtual camera position, it is possible to adjust the rendering attributes, such as visibility and transparency, of every single part in the scene. Moreover, background colour and scene lighting can be set individually for every view. The additional views can later be selected either from a drop down list in the tool bar that is associated with the activated 3D object in the Reader or from the context menu of the 3D object.

The file <views file> is structured into view sections, one for every view:

```
VIEW[=<optional name>]
  COO=<coo_x> <coo_y> <coo_z>
  C2C=<c2c_x> <c2c_y> <c2c_z>
  ROO=<roo>
  AAC=<aac>
  ROLL=<roll>
  BGCOLOR=<r> <g> <b>
  RENDERMODE=<render mode>
  LIGHTS=<lighting scheme>
  PART=<part name (required) as in the Model Tree>
    VISIBLE=true | false
    OPACITY=<part opacity>
    RENDERMODE=<see option ‘3Drender’ for possible values>
  END
  PART=<...>
  ...
  END
  etc.
END
VIEW
...
END
etc.
```

A view section starts with the keyword **VIEW**, optionally followed by a name for the view, and ends with the keyword **END**. If no name is given to the view, a default one is created, consisting of ‘View’ followed by the number of the current **VIEW** section in the file. A **VIEW** section contains optional entries for setting the camera position and global rendering attributes of the scene as well as **PART** subsections for setting rendering attributes of parts individually. Table 1 lists the entries in a **VIEW** section. Part sub-sections are opened by **PART=<part name>** and closed by **END**. There may be as many part subsections as there are parts a 3D object is composed of. Table 2 lists the possible entries in a **PART** sub-section. All entries are optional. **<part name>** is required and must match the part name as indicated in the Model Tree of the 3D object (accessible via right-click onto the model in the Reader). In order to avoid trouble it is recommended that part names be exclusively composed of ASCII characters. You may need to load the U3D/PRC model into the authoring application and edit the part names accordingly.

Table 1: Entries in a VIEW section

key	type	value
COO	three numbers	centre of orbit, see option ‘3Dcoo’
C2C	three numbers	centre of orbit to camera vector, see option ‘3Dc2c’
ROO	number	radius of orbit, see option ‘3Droo’
AAC	number	camera aperture angle, see option ‘3Daac’
ROLL	number	camera roll, see option ‘3Droll’
BGCOLOR	three numbers	canvas background colour (RGB), see option ‘3Dbg’
RENDERMODE	string	render mode of the 3D object, see option ‘3Drender’
LIGHTS	string	lighting scheme, see option ‘3Dlights’
PART	sub-section	part name as in the model tree, see Table 2 for list of possible entries

The views file can be commented. As usual with L^AT_EX, comments start with the percent sign.

To facilitate the creation of a views file, a `\movieref` link with option ‘3Dgetview’ can be temporarily inserted into the document. When clicked, it outputs a complete VIEW section corresponding to the current view of the 3D object in the Reader GUI, including camera position as well as all part and viewing options that may be set via the context (right-click) menu of the 3D object. See section 5.3 for details.

`3Djscript=<JavaScript file>`

Things like animation, lighting, background etc. may also be script driven. Option ‘3Djscript’ associates `<JavaScript file>` with the 3D object. The script will be executed upon activation of the object. Refer to the Acrobat 3D JavaScript Reference [4] for details. Directory ‘doc/javascript’ contains JavaScript example files for animation and rotation control. The files work off-the-shelf with any 3D file and may be concatenated to combine their effects.

`3Dresource=<resource file 1>[, 3Dresource=<resource file 2> [, ...]]`

Embeds additional 3D or *rasterized* image files that can be used as resources while rendering the 3D artwork. Possible file types are U3D, Postscript (L^AT_EX + `dvips`) and PDF/JPEG/PNG (pdfL^AT_EX). Embedded resources must be loaded

Table 2: Entries in a PART sub-section

key	type	value
VISIBLE	boolean	a flag (<code>'true'</code> or <code>'false'</code>) indicating the visibility of this part
OPACITY	number	a number between 0.0 and 1.0 specifying the opacity of this part
RENDERMODE	string	rendermode of this part, overrides global RENDERMODE value in parent VIEW section, see option <code>'3Drender'</code>

by the JavaScript method

```
Resource('pdf://<resource path>')
```

where `<resource path>` stands for the path to the resource file as specified by the `'3Dresource'` option. (This is just a naming convention; the files are physically embedded in the final PDF.) The following JavaScript loads an image file that was attached by `'3Dresource=images/sunset.jpg'` as the background.

```

sunset = new Image(new Resource('pdf://images/sunset.jpg'));
reh = new RenderEventHandler();
reh.onEvent = function(event) {
    runtime.removeEventHandler(this);
    event.canvas.background.setImage(sunset);
}
runtime.addEventHandler(reh);

```

5.3 Media hyperlinks

A movie, sound or 3D object may serve as the destination of hyperlinks, which are inserted into the document using the command

```
\movieref[<options>]{<label spec>}{<text>}
```

It makes `<text>` a hyperlink to the media that has been labelled with `<label spec>` using the `'label'` option of the `\includemovie` command. Media hyperlinks can be inserted at any location within the document. Clicking onto the hyperlink causes the Reader to open the page containing the media it is pointing to. The primary purpose of media hyperlinks, however, is to provide a means to control the playback.

In 2D context, clicking onto the hyperlink causes the media to stop, pause or resume, or to restart at different settings, such as frame rate, volume, starting and ending positions, depending on the `<options>` being in effect.

Options that control the playback *characteristics* are: ‘controls’, ‘endat’, ‘mouse’, ‘palindrome’, ‘rate’, ‘repeat’, ‘startat’ and ‘volume’. Their meaning is the same as for the `\includemovie` command and can be looked up in section 5.1. Another option, ‘default’ causes the media to play at the original settings which were in effect during media inclusion. These options can be used in combination with the ‘play’ option only.

The options which follow let the hyperlink change the playback *state* of media. If none of them is specified, ‘play’ is assumed.

play

Immediately restarts playback. If new playback settings have been specified through the options given above, they override the original settings being in effect during media inclusion.

stop

Stops and rewinds the media to its beginning or to the ‘startat’ position.

resume

Resumes paused media. In fact, this option makes the hyperlink a toggle switch: repeated clicking switches between paused and resumed state.

pause

Has the same effect as option ‘resume’. Added for completeness only.

close

Immediately closes the media player.

In 3D context, `\movieref` can be used to define an additional view of the object, to access a particular view from the list of predefined views or to run a JavaScript. Moreover, a link can be created for calculating optimal camera settings ‘3Droo’ and ‘3Dcoo’, instead of finding them manually. The following options are recognized:

`3Dviewindex=<index>`

Go to a predefined view of the 3D object. <index> can take ‘F’, ‘L’, ‘N’ or ‘P’ to access the first, last, next or previous element of the list of additional views, see option ‘3Dviews’ of `\includemovie`, or an integer specifying an index into the list. In the case of ‘N’ and ‘P’, repeated clicking onto the hyperlink allows to cycle through the list in forward or backward direction. ‘D’ gives access to the default view.

`3Daac=<angle>`

`3Dc2c=<x> <y> <z>`

`3Dcoo=<x> <y> <z>`

`3Droll=<roll>`

`3Droo=<r>`

Instead of referencing an existing view of the 3D object, a new one can be defined using any of these options. See section 5.2.

`3Dcalculate[=<aac angle>]`

Mainly used during document authoring. Creates a link for calculating optimal ‘3Droo’ and ‘3Dcoo’ settings of the virtual camera, which may be used to define a default view. Clicking the link opens a dialogue box from which the settings can be copied to the clipboard for later insertion into the option list of `\includemovie`. The parameter `<aac angle>` which is the camera aperture angle `<AAC` used for calculation is optional. If omitted, the one set by `\includemovie` option ‘3Daac’ is taken.

`3Dgetview`

Mainly used during document authoring. Camera settings as well as part and scene rendering attributes that correspond to the current view are printed to a dialogue box. The output is a readily formatted `VIEWS` section to be inserted into or appended to a file of predefined views. See option ‘3Dviews2’.

`3Djscript=<JavaScript file>`

Runs the script `<JavaScript file>` after clicking the link. Unlike the script that has been associated with the 3D annotation during object embedding, this JavaScript is not directly run by the JavaScript engine of the 3D plug-in, but is run by the Reader’s own scripting engine. However, full access to the API of the 3D script engine is provided through the ‘`context3D`’ property of the ‘`Annot3D`’ object. For convenience, an associative array ‘`annot3D`’ has been provided for use within `<JavaScript file>`, in order to easily access the ‘`Annot3D`’ object of the annotation the enclosing link is pointing to. The particular element of the array is referenced by the annotation’s label `<label spec>`, enclosed in quotation marks. As an example, the ‘`scene`’ object can be referenced within `<JavaScript file>` by

```
annot3D['<label spec>'].context3D.scene
```

For details about Acrobat JavaScript and its `Annot3D` object, see [5].

5.4 Compatibility commands

Two user commands have been provided that make ‘`movie15`’ a replacement for the ‘`multimedia`’ package which is part of Till Tantau’s ‘`Beamer`’ Class:

```
\movie[<options>]{<poster text>}{<media file>}
```

```
\hyperlinkmovie[<options>]{<label spec>}{<text>}
```

All of the optional arguments to `\includemovie` and `\movieref` can be used with the compatibility macros as well. In addition, they honour the following options:

```
height=<height of display>
```

```
width=<width of display>
```

Set the horizontal and vertical dimensions of the media display area (`\movie` only).

```
autostart
```

The same as ‘`autoplay`’.

`loop[=<repeats>]`

The same as `'repeat'`.

`once`

The same as `'repeat=1'`.

`showcontrols`

The same as `'controls'`.

`start=<offset in seconds>s`

Start playback at the specified temporal offset. The trailing `'s'` is mandatory.

`duration=<duration in seconds>s`

The duration of the media segment to be played. The trailing `'s'` is mandatory. Note that `'duration'` cannot be used together with option `'startat'` from `\includemovie`. In order to define a media segment options can be combined as follows: `'startat'/'endat'`, `'start'/'endat'`, `'start'/'duration'`.

6 Examples

1. A short circular MPEG movie, taken from <http://www.linux-video.net/Samples/>

```
\includemovie[
  poster,
  text=(random.mpg),
  mouse,
  repeat
]{
  (random.mpg)
  .5\linewidth
}{
  .375\linewidth
}{random.mpg}
```

2. Another MPEG movie, loaded on the fly from <http://www.linux-video.net/Samples/>. Hence, a working Internet connection is required. This time, we force the QuickTime plug-in to be used instead of MediaPlayer (Windows boxes only). Well, the latter seems to be bugged. Quod erat expectandum.

```

\includemovie[
  poster,
  label=alien,
  text=(AlienSong.mpg),
  url,
  playerid=AAPL_QuickTime,
  repeat
]{
  .5\linewidth
}{
  .375\linewidth          Slow Normal Fast Play/Pause Stop
}{http://www.linux-video.net/Samples/Mpeg1/AlienSong.mpg}
\movieref [rate=0.5]{alien}{Slow}
\movieref [default]{alien}{Normal}
\movieref [rate=2]{alien}{Fast}
\movieref [pause]{alien}{Play/Pause}
\movieref [stop]{alien}{Stop}

```

3. Embedded U3D file. It is based on a VRML model by Peter Whitehouse, <http://www.wonko.info/vrml/index.htm>. Conversion to U3D has been made with DeepExploration[®][3]. The file 'dice.vws' provides predefined views to be selected from the 3D toolbar or by right click.

```

\includemovie[
  poster,
  toolbar,
  label=dice,
  text=(dice.u3d),
  3Droo=27,
  3Dlights=Cube,
  3Djscript=turntable.js,
  3Dviews2=dice.vws
]{
  .5\linewidth
}{
  .5\linewidth
}{dice.u3d}
\movieref [
  3Dviewindex=N
]{dice}{Click here} ...      Click here to cycle through the list of
\movieref [
  3Dcalculate=60
]{dice}{This link} ...      predefined views. This link calculates
                              3D option settings for a 60° aperture
                              angle of the virtual camera.

```

A 3D quick-start guide

1. Insert the 3D object with default camera settings and a `\movieref` referencing it:

```

\includemovie[poster, label=my_label]{
    .5\linewidth
}{
    .5\linewidth
}{
    my_file.u3d
}\
\movieref[3Dcalculate]{my_label}{Click here!}

```

2. Compile and recompile the document until no more ‘movie15’ related warnings appear.
3. Open the PDF document in Adobe Reader and go to the page containing the 3D object. Click the link and wait for a dialogue box to pop up. Optionally, drag the object with the mouse to change the viewpoint of the camera (the dialogue must be closed beforehand) and click the link again.
4. Copy the settings (3Droo=..., 3Dcoo=..., etc.) from the dialogue box into the option list of \includemovie. Remove the link from the document source:

```

\includemovie[
    poster, 3Droo=33.3333, 3Dcoo=1.2345 9.8765 0
]{
    .5\linewidth
}{
    .5\linewidth
}{
    my_file.u3d
}

```

5. Again, compile and recompile the document until no more ‘movie15’ related warnings appear.

Optional steps:

6. Create a file with predefined views of the 3D object and attach it to the latter using the ‘3Dviews2’ option:

```

\includemovie[
    ..., label=my_label, 3Dviews2=my_views.vws
]{
    .5\linewidth
}{
    .5\linewidth
}{
    my_file.u3d
}

```

The views file can easily be populated using a temporarily inserted \movieref link:

```

\movieref[3Dgetview]{my_label}{Click here!}

```

Manipulate the 3D object using the mouse and any of the ‘Part’ and

‘Viewing’ options from the 3D context menu in the Reader. Visibility, lighting, transparency, render mode etc. of individual parts or of the object as a whole can be changed this way. Sometimes you may wish to move the camera target into the centre of a single part or of a group of visible parts. The context menu items ‘Part Options→Zoom to Part’ and ‘Part Options→Fit Visible’ can be used for this purpose.

When you are done, click the link to get the VIEW section, readily formatted for insertion into the views file. Repeat this procedure to get any number of views you want to predefine. The views file can be edited manually to give meaningful names to the views (change the value of the VIEW entry) or to further tweak camera settings, opacity etc.

If you are satisfied with the predefined views in the views file, the default view first specified through the options of `\includemovie` can be deleted. The first view in the views file becomes the default view then.

7. Associate a JavaScript with the 3D object:

```
\includemovie[
  ..., 3Djscript=my_script.js
]{
  .5\linewidth
}{
  .5\linewidth
}{
  my_file.u3d
}
```

JavaScript is *not* required to make use of *embedded* keyframe animation. Just click the ‘Play’ button in the 3D toolbar of the 3D annotation. However, the example file ‘`animation.js`’ in the ‘`doc/javascript`’ directory provides additional controls (accessible via the context menu of the 3D annotation) that can be used to change the speed of a running animation. The ‘Up’, ‘Down’ and ‘Home’ keys of the keyboard can also be used.

Another useful example file you may want to experiment with is ‘`turntable.js`’. It improves the rotational behaviour of the 3D object, because it prevents the object from tilting to the side while dragging the mouse.

All files in ‘`doc/javascript`’ work off-the-shelf and can be copied into a single file in order to combine their effects.

B Media formats

Table 3: Media formats for use with Adobe Reader (partially from [1])

Extension	MIME type	Description
aif, aifc, aiff	audio/aiff	Audio Interchange File Format
au, snd	audio/basic	NeXT/Sun Audio Format
avi	video/avi	Audio/Video Interleaved (animated) GIF
gif	image/gif	(animated) GIF
mid, rmi	audio/midi	Musical Instrument Digital Interface
mov, qt	video/quicktime	Apple QuickTime
mp3, m3u	audio/x-mp3	MPEG Audio Layer-3
mp4	video/mp4	MPEG-4 Video
mpeg, mpg	video/mpeg	MPEG-1 & 2 Video
prc	model/prc	Adobe PRC Format (3D), requires Reader-8.1 or higher
ra	audio/vnd.rn-realaudio	Real Audio
rm, rv	application/vnd.rn-realmedia	Real Media (video & sound)
smil	application/smil	Synchronized Multimedia Integration Language
swf	application/x-shockwave-flash	(Adobe?) Macromedia Flash
u3d	model/u3d	Universal 3D File Format
wav	audio/wav	MS Audio Format
wma	audio/x-ms-wma	Windows Media Audio
wmv	video/x-ms-wmv	Windows Media Video

References

- [1] Adobe Systems Inc.: *PDF Reference, fifth edition: Adobe Portable Document Format version 1.6*, 2004, Implementation note 144, Appendix H.3, available at http://www.adobe.com/devnet/pdf/pdf_reference.html
- [2] <http://www.3dif.org/>
- [3] <http://www.righthemisphere.com/products/dexp/>
- [4] Adobe Systems Inc.: *JavaScript for Acrobat® 3D Annotations API Reference*, 2006, available at <http://www.adobe.com/devnet/acrobat/javascript3d.html>
- [5] Adobe Systems Inc.: *JavaScript for Acrobat® API Reference*, 2006, available at <http://www.adobe.com/devnet/acrobat/javascript.html>