Automated DVD menu authoring with pdfLaTeX

Péter Szabó
Budapest University of Technology and Economics,
Dept. of Computer Science and Information Theory,
H-1117 Hungary, Budapest, Magyar tudósok körútja 2.
pts (at) cs dot bme dot hu
http://www.inf.bme.hu/~pts/

Abstract

dvdauthor is an excellent low-level free tool for video DVD authoring on Unix systems. However, it doesn't provide a convenient way for drawing the menu background and buttons. We present dvdmenuauthor, a collection of scripts for automated DVD authoring with menus. dvdmenuauthor uses pdfLaTeX macros for menu composition, Xpdf for menu rendering, and dvdauthor for DVD filesystem authoring.

1 Background

DVD-Video [8] is today's most popular home entertainment video format. Video shops and video rental services used to provide films on VHS cassettes in the 1990s, but now they offer DVD discs almost exclusively. DVD not only provides cheaper reproduction costs, better video and audio quality than VHS, and multiple camera angles, audio tracks and subtitles, but it also has an advanced, programmable (but optional) navigation facility called DVD menus (or DVD extras).

The remote control of a DVD player device has several menu buttons (such as *menu*, *top menu*, *audio*, *subtitle* and *angle*), which, when pressed, suspend playback and jump to a menu. A menu is a single-page interactive part of the DVD, designed and programmed by the DVD creator. It can be animated (possibly in a loop), and it can have audio as well. A menu has several on-screen buttons, one of them being highlighted. The arrow buttons (*up*, *right*, *down* and *left*) on the remote control can be used move the highlight, or, when the DVD is played on a computer, the highlight is moved to the button under the mouse. The *enter* button can be used to execute the action associated to the highlighted on-screen button. Possible actions:

- resume or start playback at a specific location;
- jump to another menu (possibly with a specific button pre-highlighted);
- change a playback-related variable (such as audio stream, subtitle language and angle);
- change an auxiliary variable (to be used later) — integer arithmetic operators are available;
- execute a conditional block (*if–then–else*).

Popular reasons for adding menus to a DVD:

- DVD menus are a good quality addon for Hollywood-style movies. Both the visual appearance and the sound of the menu is in theme with the movie, and the first minute spent on navigating the menu (mostly in order to select the audio stream and subtitles) is now part of the fun the spectator experiences.
- If the DVD contains a lot of material (up to 8 hours are feasible using double-layer discs and lossy compression), spectators expect an order in which they can easily find the title they are looking for. Menus with thumbnail images and title captions make navigation easier. It is also possible to have multiple menus that point to the same set of titles, but in different logical order. Usually 2×2 or 3×2 thumbnails are displayed in a single menu, and such menus are linked together using buttons. Most DVD authoring software provide an automated wizard for generating thumbnailed menus of this kind.
- DVD menus make it possible to present an interactive show to the spectators, in which they can choose among 2 or 3 endings of the movie, or they can even choose in the middle how the story should advance. Of course, movie creators must record all possible storylines, which is a lot of extra work, and the capacity of the DVD disk also limits the available choices. However, it can be feasible to give the spectator 3 choice points and thus have $2 \cdot 2 \cdot 2 = 8$ storylines altogether in a 1-hour long movie.
- It is also possible to offer a trivia game (playable by the spectator) in DVD format. For example, the famous *Who wants to be a millionaire* TV

game has a DVD version [16], in which the next question is chosen randomly from about 1000 pre-recorded questions. Even the three lifelines are present. All these are programmed as a set of DVD menus.

The process of designing and creating a DVD-Video disc is called *DVD authoring*. It consists of these steps:

1. *DVD stream authoring:* The video, audio and subtitle streams are created, imported and multiplexed together to DVD-compatible MPEG-2 program stream files. The DVD standard imposes quite a lot of restrictions on the file format, the video resolution, the frame rate, the video codec, the audio codec and the audio sample rate. However, there are tools (such as the free DeVeDe [2]) which can convert any stream to a DVD-compatible stream. Most video editing tools have DVD-compatible export filters. For simple MPEG video editing, we recommend MPEG Video Wizard [17], which is not only efficient to use, but it also runs quickly enough even in virtualised environments.
2. *DVD menu authoring:* The menu background images (or animations), buttons and captions are designed, menus and titles (i.e. streams authored in the previous steps) are combined using programmed actions. This step is the integration part of the DVD authoring process, because the way individual background images, thumbnails, captions and stream files are combined together is specified in this step. If dvdauthor [3] is used in the next step, the details of the integration are specified in its XML project input file.
3. *DVD filesystem creation:* The various stream files and declarations are combined to a DVD-Video filesystem (with the `VIDEO_TS` folder). This is a completely automatic process (and takes about 5 minutes on modern PCs for a single-layer full DVD). On Linux, dvdauthor [3] is the only well-known free tool that can do the job; other programs are usually easy-to-use frontends to dvdauthor.
4. *DVD image creation:* An ISO image file is created automatically from the DVD filesystem. On Linux systems, it is usually done with mkisofs [13], with the `-dvd-video` option.
5. *DVD disc burning:* The ISO image file is automatically burnt to a DVD disc. On Linux systems, growisofs [10] is a convenient command-line tool to do the job. It can also combine this step with the previous one (DVD image creation), so a multi-gigabyte temporary ISO image file doesn't have to be created.

2 Motivation

This article focuses on DVD menu authoring, i.e. adding menus and integrating DVD-Video components. It presents a solution based on the combination of dvdauthor XML integration and LaTeX typesetting. The reasons why such a solution can be useful:

- Our solution uses only free software and runs on Unix systems. We have tried several tools [3, 14, 7, 11], but we haven't found such a tool for Unix which is user-friendly, well-integrated (doesn't need a specific version of several dozen other software packages to work), reliable (no random crashes) and ready for production use (no major bugs and annoyances, no memory leaks). We decided to develop our own software, which is practical and usable for menu-based DVD authoring.
- Most popular video editing programs provide only a wizard, which streamlines creating simple menus (such as thumbnail buttons for each title), and doesn't let the user specify the exact menu structure.
- A template-based, non-WYSIWYG solution is useful for repetitive, automated menu generation, such as generating a navigable DVD slideshow from a set of images, or generating several DVDs (with different video content) using one menu theme.
- LaTeX provides a separation of text and design that is versatile enough for several designs to be tried (and possibly customised) for the same menu structure. Most WYSIWYG DVD menu creation tools let the user manipulate the design of one object a time. Most of them don't support requests like "let's see the same design with 10% larger buttons", and even those that support it, won't be able to adjust the spacing properly around the resized buttons. With LaTeX, however, those kinds of changes can be easily done with glue nodes and a little macro programming.
- TeX can typeset textual labels of high quality. Most DVD authoring programs have very limited typographic capabilities, for example they don't support manual line breaking, line justification, automatic line breaking, pair kerning and accented characters are not available. Using LaTeX we get all these features.

3 Design decisions

It was our intention to use existing software if possible, and add or change things where existing software is not powerful enough. We have found that dvdauthor's XML project file provides an efficient and precise way for DVD menu authoring — except for drawing the menus (and converting them to a format that dvdauthor understands). Thus we decided to supplement the XML project file with drawing operations, and write some scripts that extract the drawing operations, typeset the menus, render them to images, convert the images, and run dvdauthor to create the DVD file system. We chose LaTeX for the markup language of drawing operations, mostly because it has powerful typesetting capabilities, and its macro language is powerful enough to implement the necessary housekeeping (e.g. which button was emitted to which page).

We wanted to keep LaTeX programming at a minimum, because LaTeX is not convenient for general data processing. Thus we use LaTeX mostly for typesetting. Perl scripts generate the document to typeset from the project XML file, and Perl scripts drive the further conversion of pdfLaTeX's PDF output to images (using the pdftoppm tool of Xpdf [19]). LaTeX macros emit some metainformation to the `.aux` file (e.g. the correspondence of PDF page numbers and DVD menu button names), which is also read by the Perl scripts.

We wanted to reuse as many LaTeX typesetting constructs as possible, thus the style file just sets the page size, the margins and the default font size, and lets the user draw the menu with LaTeX. We don't enforce any specific layout, any layout can be designed using TeX boxes, glues and macro programming. However, we don't use automatic page breaks: the user has to decide in advance how many menus to have. (Automatic page breaks wouldn't fit with dvdauthor's project file easily anyway.) The style file also provides some drawing primitives useful for DVD menus: colourful frames, framed buttons, single-colour buttons (of any shape) and absolute positioning.

We designed the project file syntax so that users don't have to type the same information twice, and data relationships are often expressed by putting related pieces close to each other. For example, it is possible to specify the thumbnail image file name as an attribute to the DVD `<button>` tag. The image file name will be passed as a parameter to the appropriate macro that draws the image.

The software we have written, dvdmenuauthor, is free to use and is available for download from [6].

4 The manual way of authoring DVDs

This section gives an introduction to DVD-Video concepts, and it also presents the pure, manual way of DVD menu authoring using dvdauthor. The way presented here is similar to typesetting documents with TeX: there are a bunch of input files, most of them being plain text files written by humans, and there are some non-WYSIWYG tools, which can be applied to the input files in the correct order to produce the desired output.

4.1 DVD without menus

DVD stream authoring is beyond the scope of this paper, so we assume that the movie is already prepared in a set of DVD-compatible MPEG-2 stream files. Unfortunately, there is no validator for this file format. If there is a problem with the file (for example, the wrong audio codec is used, or the multiplexing packet size is incorrect), dvdauthor will usually complain, but the error message doesn't always indicate clearly the reason for the problem. The free video conversion tools MEncoder [12] and FFmpeg [9] can generate a conforming MPEG-2 stream if called with the proper parameters. See the source code of DeVeDe [2] for parameters to MEncoder.

A DVD-Video disc consists of titles and menus. A title is a stream that contains video and audio (multiple video and audio channels possible). The playback of a title can start at the beginning or at any specific position (given by a time offset from the beginning). A chapter is a logical unit within a title. DVD players usually let the user choose a title (by its number) to start playback at (not all players expose chapter boundaries within the title to the user).

The simplest, completely automatic way to create a DVD without menus is to use dvdauthor [3]. For example,

```
dvdauthor -o dir -t a.mpg b.mpg c.mpg
dvdauthor -o dir -t d.mpg e.mpg
dvdauthor -o dir -T
```

creates a DVD with two titles. Title 1 contains 3 chapters (from the contents of video files `a.mpg`, `b.mpg` and `c.mpg`, respectively). Title 2 contains 2 chapters (from the contents of video fies `d.mpg` and `e.mpg`, respectively). For each title, files `dir/VIDEO_TS/VTS_NN_*` are created, where `NN` is the number of the title. The last command (with the `-T`) creates the table of contents (to files `dir/VIDEO_TS/VIDEO_TS.*`). Please note that the contents of the video files are copied, so running dvdauthor takes time and needs free disk space (about the same size as the total size of the input files). The DVD filesystem created in `dir` can be played with most media players

on Linux (e.g. Xine, VLC, Kaffeine and MPlayer). If it looks right, it can be burnt to disc:

```
growisofs -dvd-compat -Z /dev/dvd \
  -dvd-video dir/
```

4.2 DVD with menus

If menus are involved or complex settings have to be specified, then a project file (in XML format) should be prepared for dvdauthor, which specifies all aspects of the DVD (how chapters, titles and menus should be formed from input files; what settings should be used; what code should be executed at events like title playback beginning, title playback end and remote control button press). Then the DVD filesystem would be created by the command

```
dvdauthor -x project.xml
```

The manual page of dvdauthor [4] gives an excellent and concise introduction to DVD-Video concepts. However, it doesn't contain examples for complex XML projects. To get such an example, one should try some GUI DVD authoring tools (such as [11, 14]) and see what files they generate.

A DVD menu is similar to a title, with some extra interactive features, such as buttons and actions. Buttons form an extra visual layer above the menu. Each button is a rectangular image (other shapes can be specified using transparent pixels) with only a small number of colors (≤ 16). Buttons on a menu page might not overlap. Each button has 4 neighbours (*left*, *right*, *up* and *down*). A neighbour is activated when the user presses the corresponding arrow button on the remote control. When a menu is being shown, it has a current button. Only the image of the current button is drawn over the video, none of the other buttons are displayed. When another button is activated, it becomes the current button, it gets displayed (and the previous button gets hidden). When the user presses the *enter* remote control button, the code associated with the current button is executed. The code can be specified in the project XML file inside the corresponding `<button>` tag. The syntax is similar to a very small subset of C, it is documented in the manual page of dvdauthor.

Menus can also contain actions. An action is like a button with code (to be executed), but without a visual representation. An action is activated either by the arrow keys on the remote control (in this case, the action must be a neighbour of the current button), or by special keys (such as *angle*) on the remote control. Actions are very briefly documented; just a little information can be found in the manual page of *spumux* (a tool which is part of the dvdauthor suite). Actions can also be used to jump to a different menu without the *enter* remote control button. [15] is a detailed tutorial about this. Animated thumbnails (where the thumbnail of the current button is animated) can be also be created this way. To have animated thumbnails, a separate, animated menu has to be created for each button, each menu having only one button and neighbouring actions, which jump to another menu.

Both titles and menus support executing code before the title or menu is entered (specify such code inside the `<pre>` in the dvdauthor XML fie) and when it is left (use the `<post>` tag). The `<pre>` tag of the main menu can contain code to skip the intro video unless the disc playback has just started. This can be implemented as a conditional jump instruction. The condition should depend on a variable, which is set just before jumping to the intro.

The similarity of titles and menus implies that menus can have audio and animation. These features for menus are provided by default, since the menu background is an MPEG-2 file itself, which can contain audio, and of course can contain animation. DVD authoring applications (including dvdmenuauthor) usually support only still images for menus, and they take care of converting these images to MPEG-2 videos of a few seconds in length. Making the menu video loop is straightforward: the menu's `<post>` code has to be extended with a jump command that jumps to the beginning of the menu. DVD players are usually slow when jumping (partially because a seek on the DVD disc is slow), so expect about a half second of audio lag when the menu loops. Some players also ignore remote control buttons during this lag.

There are some additional concepts which are important to understand before designing a DVD structure by hand. Such concepts are: vmgm, titleset and cell. These are documented in the manual page of dvdauthor [4].

Sometimes a code snippet in a DVD program is too long. Unfortunately, dvdauthor doesn't always indicate this error condition properly. The solution is to split the containing menu to two or more menus, each of them containing half of the code, and jumping to each other when necessary.

Restrictions There are some restrictions on the sources and targets of direct jumps in the DVD program code. (For example, one cannot jump from one titleset directly to another one. Another example: in some jumps, the target menu number cannot be specified — the so-called entry point must be given.) To overcome these restrictions, an indirect jump can be used: set a variable, jump to the main menu, whose

<pre> code examines the variable and jumps to the desired target title or menu. dvdauthor provides a scarcely documented facility (the jumppad="1" attribute) to do this automatically. However, this might produce extra errors if the DVD program code is long. It is safer to implement the indirect jumps by hand.

4.3 Drawing the menus

All aspects of the DVD can be specified straightforwardly in the dvdauthor XML project file, except for the menu background image, the menu button images and the button neighbour relationships. Our aim with dvdmenuauthor was to automate this process as far as possible, while still leaving full control to the user. But first let's see a manual method. The visual part of a DVD menu consists of:

- *background stream:* an MPEG-2 stream with video and audio. For simple menus, the audio is silent, and the video contains a single still image repeated for a couple of seconds. (dvdmenuauthor supports only still images without audio.)
- *button highlight layer:* a single image with a few colours and transparency. This layer consists the union of the button images. When the DVD player displays a menu, it draws the image of the current button (taken from the button highlight layer) over the background stream. A simple button highlight layer contains only a single colour besides the transparent pixels. (dvdmenuauthor supports only a single colour.)
- *button select layer:* similar to the button highlight layer, but a button is drawn from here when it is activated (with *enter*). The duration that the image is displayed is just a few hundred milliseconds: it lasts until the DVD player loads the next title or menu. Usually the button highlight layer is the same as the button select layer, but with a different color.
- *button bounding boxes:* the rectangular bounding box of each button on the menu. These boxes must not overlap.
- *button neighbours:* the name of the left, right, up and down neighbour for each button. dvdauthor is able to infer neighbourhood relationships from bounding box coordinates.
- *button and action names:* these are used by dvdauthor to identify the button (or action) within the menu, in order to be able to add code to be executed when the button is selected.

It is quite cumbersome to keep all these visual elements in sync by hand when drawing the menus. For example, if we move or resize a button, then the background stream, the button layers and the button bounding boxes have to be properly modified. dvdmenuauthor does all these automatically.

To find out how to assemble the visual elements to an MPEG-2 stream, the easiest way is to examine the auxiliary files generated by GUI frontends to dvdauthor [11, 14]. The XML syntax is explained in the manual page of dvdauthor and its *spumux* tool, and also in [5]. dvdmenuauthor contains a Perl script called *genmpeg.pl*, which can generate a DVD-compatible MPEG-2 stream from a series of still images.

5 DVD menus with dvdmenuauthor

5.1 A menu with thumbnails

Figure 1 shows a typical thumbnail menu in a 3×2 layout. The menu has a background, a title caption, up to 6 thumbnail buttons (now actually 5), a caption for each thumbnail, and three navigation buttons to reach other menus. For simplicity, big numbers are displayed instead of real thumbnails from the video. In the figure, thumbnail button number 1 is highlighted with an ochre rectangular frame.

Figure 2 shows how to define such a menu in dvdauthor's XML project file. All the visual elements, including the background, the button layers and the button bounding boxes are encoded in the file *menu42.mpg*. Probably *spumux* was used to multiplex these visual elements to the file. The figure shows that each button has a name and a corresponding program code to execute when the button is activated. If there are any actions in the menu, they also appear as <button> tags here.

Figure 3 shows how to draw the same menu using dvdmenuauthor. It also illustrates the following features of the software:

- LaTeX markup can be used to typeset the captions (see \emph).
- TeX's line breaking algorithm can be used (see caption of button 1).
- The visual design is separated from the menu-specific data (actual captions and thumbnail images) using templates. Only the menu-specific data is shown on the figure.
- All information needed to render a button are packed together to the <button> XML tag. The attributes with the tex: namespace are passed to LaTeX.
- There is no need to specify button bounding boxes.

To further illustrate the magic happening, Figure 4 shows the LaTeX code snippet generated from

Figure 1: A menu created by dvdmenuauthor

```
<pgc>
  <vob file="menu42.mpg" pause="inf" />
  <button name="e1"> g5=6; </button>
  <button name="e2"> jump menu 2; </button>
  <button name="e3"> jump menu 3; </button>
  <button name="e4"> jump menu 4; </button>
  <button name="e5"> jump menu 5; </button>
  <button name="prev"> jump vmgm menu 1; </button>
  <button name="back"> jump vmgm menu 1; </button>
  <button name="next"> jump vmgm menu 1; </button>
</pgc>
```

Figure 2: A menu definition in the dvdauthor project file

```
<pgc>
  <tex:prepage>
    \thispagebgimage{}{pal_bg_light}
    \thispagetemplate{palthumbsix}
    \menucaption{Fazekas szalagavató 1998.\ december}
  </tex:prepage>
  <vob tex:file="" pause="inf" />
  <button name="e1" tex:image="1.png" tex:caption="a szalagtűzés \emph{előtt}"> g5=6; </button>
  <button name="e2" tex:image="2.png" tex:caption="szalagtűzés"> jump menu 2; </button>
  <button name="e3" tex:image="3.png" tex:caption="osztálytáncok"> jump menu 3; </button>
  <button name="e4" tex:image="4.png" tex:caption="egyéb táncok"> jump menu 4; </button>
  <button name="e5" tex:image="5.png" tex:caption="videófelvételek"> jump menu 5; </button>
  <button name="prev" tex:dummy=""> jump vmgm menu 1; </button>
  <button name="back" tex:dummy=""> jump vmgm menu 1; </button>
  <button name="next" tex:dummy=""> jump vmgm menu 1; </button>
  <post> jump vmgm menu 1; </post>
</pgc>
```

Figure 3: A menu definition in the dvdmenuauthor project file

```
\begin{dvdmenupage}{e1,e2,e3,e4,e5,prev,back,next}
\thispagebgimage{}{pal_bg_light}
\thispagetemplate{palthumbsix}
\menucaption{Fazekas szalagavató 1998.\ december}
\begingroup\def\dvdbuttonattrXname{e1}
  \def\dvdbuttonattrXcaption{a szalagtűzés \emph{előtt}}
  \def\dvdbuttonattrXimage{1.png}
\dvdprocessbutton\endgroup
\begingroup\def\dvdbuttonattrXname{e2}
  \def\dvdbuttonattrXcaption{szalagtűzés}
  \def\dvdbuttonattrXimage{2.png}
\dvdprocessbutton\endgroup
...
\begingroup\def\dvdbuttonattrXname{next}
  \def\dvdbuttonattrXdummy{}
\dvdprocessbutton\endgroup
\end{dvdmenupage}
```

Figure 4: The LaTeX code snippet generated from the dvdmenuauthor definition

the dvdauthor definition on Figure 3. We can see that for each attribute with a `tex:` prefix, a macro `\dvdbuttonattrX...` is defined, and the command `\dvdprocessbutton` is called after the macro definitions for each button. The `tex:dummy` attribute is just an indicator that the button must be processed by LaTeX.

The macros `\menucaption` and `\dvdprocessbutton` are defined in template *palthumbsix*. They take care of the visual formatting of the menu data. `\dvdprocessbutton` has the available button names hardwired, and it formats and positions a button based on its name.

5.2 New LaTeX commands provided

Although dvdmenuauthor encourages the use of existing LaTeX commands, it also defines some new commands, most of them related to positioning and button typesetting.

`\thispagecolor{`*colorname*`}`

Changes the background of the current page to the specified color. Like most graphics operations in dvdmenuauthor, it works only with pdfLaTeX.

`\thispagebgimage{`*optionlist*`}{`*filename*`}`

Sets the background image for the current page. The image will be loaded by the `\includegraphics` command using the *pdftex* driver.

`\thispagetemplate{`*templatename*`}`

Selects the specified template for the current page. This means defining some macros, for example the sample template *palthumbsix* defines `\menucaption` and `\dvdprocessbutton`.

`\putat{`x`}{`y`}`*boxspec*

Typesets the specified material with its reference point at (x, y). Creates a properly shifted box of size zero. Can be used for absolute positioning if called at the top of the page. Can be used with `\vbox` for last line alignment or `\vtop` for first line alignment. This command can be used in templates.

`\begin{dvdmenupage}{`*buttonlist*`}`

Renders a DVD menu page with the specified buttons (in the specified order). For technical reasons (see Subsection 6.2) each button is rendered on its own onto a separate PDF page.

`\framehbox{`*sep-dimen*`}{`*hbox-contents*`}`

Similar to the built-in command `\framebox`, but allows catcode changes and verbatims in the box. Unfortunately, catcode changes don't work in general in dvdmenuauthor, because the *dvdmenupage* environment reads its contents to a macro for multiple rendering.

`\aliascolor{`*oldname*`}{`*newname*`}`

Copies a colour definition to be usable as a different name.

`\dvdtextbutton{`*name*`}{`*text*`}`

Typesets a button. Doesn't wrap its contents in a box, so a button can be in a middle of a paragraph, even line breaks are allowed. Each button has three forms: background, highlighted and selected. In background mode, *text* is typeset normally, in highighted mode, it is typeset in *dvdhighlightedcolor* (without color changes and images), and in selected mode it typesets in *dvdselectedcolor*.

`\dvdframebutton{`*name*`}{`*text*`}`

Typesets a button inside a `\hbox`. In background mode, emits the `\hbox`. In highlighted mode, it emits an empty box surrounded by a frame (using the parameters `\dvdbuttonframesep`, `\dvdbuttonframewidth` and *dvdhighlightedcolor*). This command can be used to typeset thumbnail buttons.

`\begin{narrowcentering}`

Like `\begin{centering}`, but doesn't reset the natural width of `\leftskip` and `\rightskip` to zero.

5.3 Working with dvdmenuauthor

TeX users are familiar with the edit–compile–preview cycle of TeX document preparation, possibly extended with a final conversion and printing or publishing. The same cycle exists with dvdmenuauthor. In the edit cycle, the user edits an XML project file with LaTeX markup for the menus, in the compile cycle the user compiles the project files (and the media files it refers to) to a DVD image, and finally the user previews the DVD on screen, including the menus and titles. The publishing step is burning the DVD filesystem to disc.

In the edit step any text editor can be used, preferably one with XML syntax highlighting facilities. Unfortunately, LaTeX snippets won't be highlighted as LaTeX markup. The dvdmenuauthor contains an example project file *ex.dmp.xml* .

The compile step consists of feeding the project file to a few scripts. The *Makefile* in the dvdmenuauthor distribution automates this. (There is no incremental compilation support yet: the whole DVD filesystem is regenerated from scratch in each `make` run.) The following steps constitute compilation:

1. *Generating the menu LaTeX source file and the dvdauthor XML project file.* This is done by the *gendap.pl* Perl script.
2. *Compiling the LaTeX source file to PDF.* This is just a regular pdfLaTeX run. The *Makefile* runs pdfLaTeX twice, in case there are `\ref`s.
3. *Rendering the PDF to PPM raster image files.* This is done by the excellent *pdftoppm* utility from Xpdf.
4. *Combining the PPM images to short MPEG-2 streams for the menus.* This is done by the *genspuxml.pl* Perl script, which calls another Perl script *genmpeg.pl* (to generate an MPEG-2 stream from still images), and the *spumux* tool from dvdauthor (to multiplex the button layers to the background). Since some image processing is done in Perl, this step can take about three seconds for each menu.
5. *Authoring the DVD filesystem.* This is just a simple *dvdauthor* run (with the `-x` option using the XML project file generated in the first step). This step might take quite a lot of time (up to 5 minutes on modern systems for a full, 4.7 GB DVD), and it needs free disk space about the same size as the sum of the input video sizes.

The recommended DVD preview application is Xine [18]. It can be installed from source in any modern Linux distribution. Although the user interface (toolbars and menus) of Xine is quite ugly, and not convenient to operate, Xine has very good keybindings, especially suitable for DVD menu navigation (a remote control panel is also available — press Alt–⟨E⟩ to make it visible). See the manual page of xine for the list of available keys. MPlayer is not recommended for DVD menu testing, because MPlayer doesn't support DVD menus yet. Although VLC has DVD menu support, sometimes it crashes or behaves unexpectedly. Kaffeine and Totem should also be given a try.

Since *dvdauthor* runs slowly when the videos to be put on the DVD are large, the compile step might be too slow for the user. To solve this, dvdmenuauthor has the *dvdmenutest.sh* shell script, which warps a dvdauthor project XML files so that all titles (but not menus!) are replaced with dummy short videos, so that the total video size would be small, and thus *dvdauthor* runs quickly.

The publishing step means creating an ISO image (with *mkisofs*) and/or burning it to disc (with *growisofs*). The `-dvd-video` flag of *mkisofs* must be specified in both command lines.

6 Implementation tricks

Some details of the implementation are worth mentioning, because the tricks employed can be useful in other TeX or DVD-related projects.

6.1 Bounding box calculation

dvdmenuauthor calculates button bounding boxes automatically. This operation is impossible within TeX, because there is no way to inquire about the absolute (x, y) coordinates of an item within a box.

Solution: dvdmenuauthor uses a bluescreen technique [1]. The LaTeX component emits each button on its own to a PDF page with a blue background, and a Perl script analyses the rendered page. The bounding box is the smallest rectangle on the page whose complement contains only blue pixels. A little extra housekeeping is done for identifying the button pages belonging to the same menu, because later they have to be merged again to a button highlight layer.

Limitations: It is not possible to force a bounding box larger than it appears. It is not possible to use that specific shade of blue in buttons. Fortunately, the colour is configurable.

6.2 Rendering only parts of the page

Each page has to be rendered many times: once for the background, once for each button highlight image and once for each button select image. While rendering the button highlights and selects, nothing except for the current button must be drawn. While rendering the background, the buttons must not be drawn. These requirements call for a facility which disables rendering parts of a page, but adjusting the spacing just as if they were there.

A simple solution would be to ask the user to put all visible material inside `\maybe{...}`, e.g.

```
\maybe{This} \maybe{is} \maybe{surely}
\maybe{an} \maybe{unclear} \maybe{caption}
```

But users may be annoyed by the tons of `\maybe`s needed to form a paragraph with line breaks.

Solution: dvdmenuauthor uses PDF coordinate system transformations to move away material that is to be skipped. A coordinate translation of (10000, 10000) is used at the beginning of the page, so all material is rendered outside the visible region by default. When one needs a specific part of the page actually rendered, a translation of $(-10000, -10000)$ is used. Since TEX's line breaking algorithm doesn't know about the translations (it treats them as `\special`s), this nicely works together with TEX glues and line breaking.

Limitations: Problems might occur when material is inside a `\rotatebox` or a `\scalebox`. That is, buttons should not be rotated. It would be possible to overcome this limitation by making `\rotatebox` and friends aware of the translation.

6.3 Single-colour rendering

`\dvdtextbutton` renders the highlighted version of the button in a different colour. Colour changes are disabled within that rendering.

Solution: All colour changes can be disabled by clearing the body of the macros `\set@color` and `\reset@color`. Image inclusions have to be disabled too, because there is no way to modify the colour of raster images included in pdfLATEX.

6.4 International character support

Accented and other international characters must survive the XML to LATEX conversions, and they should be typeset by LATEX properly.

Solution: The character set of the XML document must be specified in the `encoding=` attribute of the `<?xml` processing instruction. This declaration is read by dvdmenuauthor, and the proper `\usepackage[...]{inputenc}` is emitted automatically. For clarity, dvdmenuauthor doesn't try to interpret the XML document as characters — its built-in XML parser just treats the project file as a sequence of ASCII-based 8-bit bytes. This ensures that accented characters don't get mangled during copying from one file to another.

7 Limitations

Sometimes the user wants pixel-accurate rendering. For example, unscaled raster images should be rendered sharply on pixel boundaries, not interpolated between half-pixels. In TEX, the bp (PostScript big point) is used for the basis of dimension calculations, to avoid rounding errors when emitting the PDF (because PDF expresses all dimensions in bp). The *pdftoppm* tool almost always renders rules and images exactly and sharply, but sometimes it renders a PDF rule of width 4 bp badly: the rule becomes 5 pixels wide in the PPM image. This issue should be investigated further, possibly modifying Xpdf or trying another rendering engine.

Since the button highlight layer may use only a few colours (≤ 16), it is not feasible to use antialiased text in this layer. However, if the background text is antialiased, but the highlight is not, then the highlight doesn't cover exactly the background, which looks ugly. If antialiasing is turned off entirely (as a command line option to *pdftoppm*), text will look noticeably uglier (depends on the font used). It is not possible to turn off antialiasing selectively.

dvdmenuauthor has been tested on full PAL (720×576) videos. The DVD standard allows several image sizes for both PAL and NTSC. Support for all possible sizes should be added to each tool in dvdmenuauthor. Other parameters (such as button neighbourhood relations) should also be made configurable.

Videos are usually rendered in a different aspect ratio than their image size. The most common aspect ratios are letterbox (4:3) and widescreen (16:9). In a full PAL DVD, letterbox implies scaling from 720×576 to 768×576, and widescreen implies scaling from 720×576 to 1024×576. Thus there is a small distortion in the letterbox case and a big distortion in the widescreen case. For practical rendering reasons, dvdauthor uses the image size (720×576) and doesn't care about the distortion made by the aspect ratio correction. Thus, when authoring widescreen DVDs, menu captions would become wider than expected.

To solve this, one could render at viewing size (thus degrading rendering quality a little) or one could condense the fonts horizontally (not easy to do in TeX, but the pdfTeX font expansion feature might be useful).

8 Conclusion

TeX can be used productively for typesetting many different kinds of work: high quality (possibly international) documents, maths, presentation slides and music, to name a few. Other uses of TeX include developing software and its documentation together, preparing web-ready books, and rearranging PDF pages. DVD menu authoring is also a new, non-standard use.

We have designed dvdmenuauthor, a program for project-file-driven (non-WYSIWYG) automated DVD menu authoring with powerful menu drawing facilities. It is now a free proof-of-concept implementation. DVD menus can be drawn using LaTeX markup to have high typographic quality output. Layout and menu data can be separated using templates. Since dvdauthor is used for integration, users have full freedom to create complex and/or smart menus they want. The system design (compilation and templates) makes it possible to experiment with layout changes any time in the authoring process.

dvdmenuauthor is not easy to start using for an average user who expects WYSIWYG and wizards. However, we hope that LaTeX users would find it convenient, and the software can evolve from its present proof-of-concept version to a stable, general and powerful tool like the mainstream TeX-based programs.

References

[1] Wikipedia article on the Bluescreen technique. http://en.wikipedia.org/wiki/Bluescreen

[2] DeVeDe, a GUI program to create video DVDs suitable for home players, from any number of video formats supported by MPlayer. http://www.rastersoft.com/programas/devede.html

[3] dvdauthor, a tool that assembles multiple mpeg program streams into a suitable DVD filesystem. http://dvdauthor.sourceforge.net/

[4] The manual page of dvdauthor. http://dvdauthor.sourceforge.net/doc/

[5] Anders Dahnielson. DVD Author Primer. December 31, 2003. http://en.dahnielson.com/2003/12/dvd-author-primer.html

[6] dvdmenuauthor, automated DVD menu authoring with pdfLaTeX. http://freshmeat.net/projects/dvdmenuauthor

[7] DVDStyler, a cross-platform GUI DVD authoring system. http://www.dvdstyler.de/

[8] Wikipedia article on DVD-Video. http://en.wikipedia.org/wiki/DVD-Video

[9] FFmpeg, a very fast command-line video and audio converter. http://ffmpeg.mplayerhq.hu/

[10] DVD+RW-tools, a set of tools for burning and examining DVD discs. http://fy.chalmers.se/~appro/linux/DVD+RW/tools/

[11] KMediaFactory, template-based GUI DVD authoring software. http://freshmeat.net/projects/kmediafactory/

[12] MEncoder, the command-line movie encoder of the MPlayer suite. http://www.mplayerhq.hu/

[13] mkisofs, a tool for creating CD and DVD filesystems. http://cdrecord.berlios.de/

[14] 'Q' DVD-Author, a GUI frontend for dvdauthor and other related tools. http://qdvdauthor.sourceforge.net/

[15] Samantha Lane. Switched menus with dvdauthor. http://www.geocities.com/samanthalane/dvd/index.html

[16] Who Wants to Be a Millionaire — DVD game review. http://www.ciao.co.uk/Who_Wants_To_Be_A_Millionaire_DVD_Game__Review_5480599

[17] Womble MPEG Video Wizard, a commercial nonlinear MPEG video editor. http://www.womble.com/products/

[18] Xine, a free multimedia player. http://xinehq.de/index.php/home

[19] Xpdf, an open source viewer for PDF files. http://www.foolabs.com/xpdf/