Presentazioni animate in LaTeX

Gianluca Pignalberi

Sommario

Le presentazioni fatte con LaTeX non sono affatto condannate alla staticità totale. Un manipolo di strumenti e le possibilità del formato PDF premettono risultati più che perfetti.

Abstract

LaTeX presentations are not condemned to stay totally static. A small pool of tools and the PDF format capabilities allow more than perfect results.

1 Introduzione

A causa della mia impossibilità a partecipare al GUIT *meeting 2009* ho dovuto ingegnarmi con la presentazione. Visto che nel meeting precedente ho causato non pochi problemi logistici affinché qualcuno presentasse in mia vece, mi serviva un modo per le presentazioni di... autopresentarsi.

La maggior parte dell'utenza informatica crede che PowerPoint sia l'unico strumento per fare presentazioni (tanto da usare la locuzione "Presentazione PowerPoint" come se il nome commerciale di un prodotto fosse assurto ad apposizione). L'utenza più smaliziata conosce addirittura Impress di OpenOffice.

Ovviamente in ogni categoria ci sono una o più sottocategorie, e la categoria degli utenti di TeX non fa eccezione. Non credo di sbagliare molto dicendo che esiste ancora qualche "dinosauro" che impagina le proprie presentazioni su pagine A4, stampate su fogli lucidi, riducendosi poi a coprire parte dei lucidi con i fogli opachi, a inseguire la "classicità". Credo anche che molti utenti usino invece una classe o un pacchetto appositamente realizzati per le presentazioni. Questo articolo è per tutti loro, in quanto dimostrerà praticamente che, anche nelle presentazioni, LaTeX:

1. è slegato dai formati di pagina usuali;
2. è in grado di fare quello che fa PowerPoint;
3. lo fa in modi diversi, restituendoci la facoltà di scelta;
4. permette ottimi risultati.

Farò tutto questo descrivendo, come caso d'uso, proprio la creazione della mia presentazione per il GUIT *meeting 2009*.

2 Una presentazione minimale con beamer

Beamer è forse la più famosa classe LaTeX per realizzare presentazioni. Nel tempo si sono successe varie classi e diversi strumenti, a partire da SLiTeX, passando per Pdfscreen e Prosper fino a PPower4. Beamer sembra riassumere tutte le caratteristiche dei citati programmi, e sembra includerne altre.

La struttura di un documento Beamer minimale è molto semplice: si parte con il solito preambolo, in cui la classe usata sarà proprio beamer e in cui includeremo tutti i pacchetti usati per compilare la presentazione e stabiliremo un tema e uno schema di colore, dopodiché passeremo al corpo del documento, che conterrà quasi certamente comandi specifici di Beamer (anche se non siamo obbligati a farlo, denotando un'idiosincrasia tra intenzioni e azioni).

Un esempio poco più che minimo, con un solo frame e dei blocchi al suo interno, è il seguente:

```
\documentclass{beamer}

\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage[italian]{babel}
\usepackage{graphicx}

% This is the file main.tex
\usetheme{JuanLesPins}
%\usecolortheme{orchid}
\title[\LaTeX{} e \emph{comma below}]{combelow: abbasso i segni diacritici di serie B}
\author{Gianluca Pignalberi}
\date{Pisa, 17 ottobre 2009}
\begin{document}
\section{Sommario}
\begin{frame}{Sommario}
\begin{block}{Sommario}
Romeno e lettone devono essere considerate lingue di serie B nel mondo di \TeX?
Se finora lo potevano essere, questo piccolo pacchetto tenta di riportarle al
livello giusto con il semplice uso del segno diacritico corretto. Niente pi\'u
cediglia al posto del comma below.
```

```
22 \end{block}
23 \begin{block}{Abstract}
24 Should Romanian and Latvian be
       considered second choice
       languages in the \TeX{}
25 world? Though up to now they
       could have been, this small
       package tries to put
26 them back at the right level,
       just using the correct
       diacritic mark. No more
27 cedilla instead of comma below.
28 \end{block}
29 \end{frame}
30 \end{document}
```

La linea 1 contiene la dichiarazione della classe usata per comporre il documento; le righe 3–6 specificano i pacchetti usati per comporre la presentazione; nella riga 9 il comando `\usetheme` indica quale tema (schema grafico) avrà il documento (la riga 10, qui commentata, contiene la scelta dello schema di colore da usare). Le righe 11–13 contengono titolo, autore e data del documento. Il titolo breve, quello tra parentesi graffe nella riga 11, sarà il titolo mostrato in alto in tutti i frame della presentazione.

La riga 15 mostra il ben noto comando `\section`: il suo contenuto viene mostrato subito sotto al titolo del documento in ogni slide entro il suo *scope* (cioè finché il compilatore non incontra un nuovo comando `\section`). Ogni comando `\section` viene inoltre usato per comporre il sommario (Table of Contents).

La riga 16 contiene l'istruzione che dichiara l'inizio di un frame (una pagina della presentazione) con il relativo titolo. Quest'ultimo è mostrato in grande subito sopra il frame vero e proprio (a seconda del tema, le posizioni qui discusse possono essere diverse). Le righe 17 e 23 dichiarano due blocchi diversi all'interno dello stesso frame. Anche questi hanno il proprio titolo, messo in una fascia colorata diversamente dal resto del blocco. Il blocco conterrà tutto il testo contenuto nel documento tra i comandi `\begin{block}` e `\end{block}`. Allo stesso modo un frame è formato da tutto quanto contenuto tra `\begin{frame}` e `\end{frame}`.

Il risultato del codice precedente è quello mostrato nella figura 1.

Un frame corrisponde all'intera area di una pagina della presentazione (quella che chiameremmo normalmente *slide*), mentre un blocco corrisponde a un pezzo di testo composto all'interno di un rettangolo colorato. Ovviamente nessuno ci vieta di scrivere del testo senza includerlo in un blocco: il testo starà sullo sfondo del frame. Nel listato seguente vediamo lo stesso esempio, ma senza usare i blocchi:

```
1 \documentclass{beamer}
```

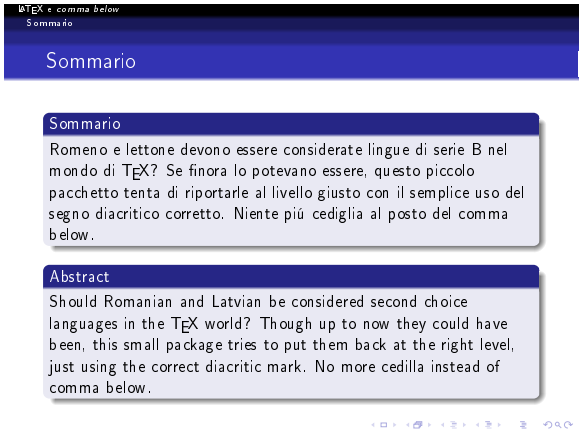

FIGURA 1: Presentazione composta da una sola pagina. La pagina è composta da un frame e relativo titolo e da due blocchi con i relativi titoli

```
2  \usepackage[T1]{fontenc}
3  \usepackage[latin1]{inputenc}
4  \usepackage[italian]{babel}
5  \usepackage{graphicx}
6  \usetheme{JuanLesPins}
7  \title[\LaTeX{} e \emph{comma
       below}]{combelow: abbasso i
       segni diacritici di serie B}
8  \author{Gianluca Pignalberi}
9  \date{Pisa, 17 ottobre 2009}
10 \begin{document}
11 \section{Sommario}
12 \begin{frame}{Sommario}
13 Romeno e lettone devono essere
       considerate lingue di serie B
        nel mondo di \TeX?
14 Se finora lo potevano essere,
       questo piccolo pacchetto
       tenta di riportarle al
15 livello giusto con il semplice
       uso del segno diacritico
       corretto. Niente pi\'u
16 cediglia al posto del comma
       below.
17
18 Should Romanian and Latvian be
       considered second choice
       languages in the \TeX{}
19 world? Though up to now they
       could have been, this small
       package tries to put
20 them back at the right level,
       just using the correct
       diacritic mark. No more
21 cedilla instead of comma below.
22 \end{frame}
23 \end{document}
```

Il documento manca solo delle righe di apertura e chiusura dei blocchi, e il risultato della sua compilazione è molto più lineare, come possiamo vedere nella figura 2.

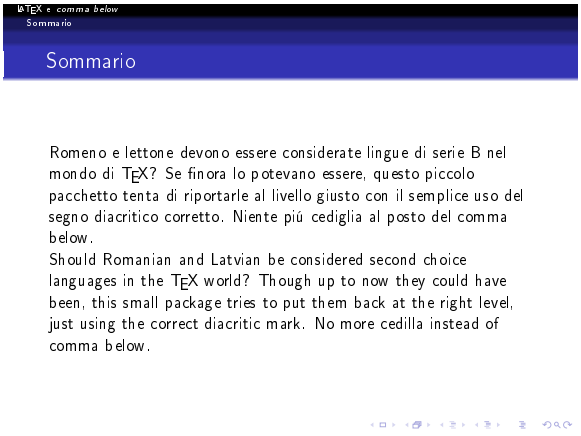

FIGURA 2: La presentazione mostrata nella figura 1 ha ora perso i blocchi, ponendo il testo direttamente nel frame

3 Il signore delle slide

Continuare a scrivere le presentazioni secondo i due basilari esempi precedenti fa sì che tutte le pagine della presentazione siano presentate nella loro interezza, una per una, senza che si possa svelare man mano il contenuto di ogni pagina.

Ovviamente Beamer ha un meccanismo per mostrare progressivamente il contenuto di una pagina. In realtà il meccanismo è più generale e serve a mostrare parti di un frame solo in determinati momenti (che non sono legati al tempo, come vedremo andando avanti), che io chiamerei *transizioni*.

Il meccanismo di cui parlo ha una sintassi molto semplice: immediatamente prima della parte di testo da mostrare si dà un comando `\onslide`, seguito dall'intervallo in cui mostrare il testo stesso. Supponiamo che un frame abbia tre frasi, che vogliamo mostrare una dopo l'altra; prima del testo della prima frase metteremo il comando `\onslide<1->`, prima del testo della seconda daremo `\onslide<2->`, e prima del testo della terza frase scriveremo `\onslide<3>`. Un numero n seguito da un trattino indica che il testo seguente verrà mostrato dalla transizione n fino all'ultima. Se dopo il trattino mettiamo un altro numero (m), questo indicherà l'ultima transizione in cui il testo sarà visibile (dopo non lo sarà più). Un unico numero n significa invece che il testo sarà visibile solo all'n-esima transizione. Infine, un trattino seguito da un numero m indica il testo presente dalla prima slide all'm-esima.

Come si evince dal nome del comando, Beamer definisce *slide* quella parte di frame che viene presentata in un dato momento. È una cosa simile al meccanismo di un cartone animato, in cui al lucido di sfondo si sovrapponevano uno o più lucidi con i personaggi e altri elementi, e un concetto del tutto uguale ai livelli delle immagini nei programmi di fotoritocco come Gimp o Photoshop.

Tutto in Beamer può costituire una slide e determinati elementi hanno una sintassi semplificata proprio per sveltire la scrittura. Nei comandi specifici di Beamer possiamo dare i numeri di slide direttamente dopo il comando, senza specificare `\onslide`. Nel listato seguente questa sintassi è usata nella riga 14; le righe 19, 22 e 25 usano `\onslide`.

```
1  \documentclass{beamer}
2  \usepackage[T1]{fontenc}
3  \usepackage[latin1]{inputenc}
4  \usepackage[italian]{babel}
5  \usepackage{graphicx}
6  \usepackage{combelow}
7  \usetheme{JuanLesPins}
8  \title[\LaTeX{} e \emph{comma
       below}]{combelow: abbasso i
       segni diacritici di serie B}
9  \author{Gianluca Pignalberi}
10 \date{Pisa, 17 ottobre 2009}
11 \begin{document}
12 \section{Introduzione}
13 \begin{frame}{Introduzione}
14 \begin{block}{Cosa dice Unicode
       }<1->
15 Unicode ci dice che le lettere
       con il segno a uncino o
       quello a virgola sono
16 sempre ``con cediglia''.
17 \end{block}
18
19 \onslide<2->
20 La forma della cediglia varia in
        base alla lettera: \c{c} vs
       .~\cb{k}.
21
22 \onslide<3->
23 \cb{T} e \cb{S} sono considerate
        varianti tipografiche per il
        romeno di \c{T}
24 e \c{S}.\\
25 \onslide<4>
26 Il simbolo `~\cb{}~' ha comunque
        il suo nome: \emph{comma
       below} (o
27 \emph{comma accent}).
28 \end{frame}
29 \end{document}
```

Il risultato della divisione in slide (in questo caso, quattro) è visibile nella figura 3.

Come già accennato, è molto facile ottenere frame in cui le slide sono visibili solo in determinati momenti, specificando in quali momenti della presentazione una slide deve essere visibile. L'effetto è di ottenere un testo che appaia quando serve e scompaia quando deve sparire, rimanendo solo il tempo necessario.

Figura 3: Un frame formato da quattro slide, mostrate una dopo l'altra

4 Tempo

Quanto visto finora implica che un umano sia posizionato al computer per "voltare pagina" ogni volta che serva. Nel mio caso, però, dovendo io essere assente dal convegno, avrei avuto bisogno di qualcuno che "voltasse pagina" per me.

Beamer è in grado di automatizzare esigenze come la mia e fare in modo che la pagina si volti da sola, o nel frame compaia una nuova slide. Il "cronometro" che temporizza i cambiamenti è dato dal comando `\transduration`. Quest'ultimo prende in input il numero di slide tra parentesi angolari e il numero di secondi tra parentesi graffe. Ad esempio, possiamo temporizzare le slide del precedente codice con

```
\transduration<1>{10}
\transduration<2>{10}
\transduration<3>{10}
\transduration<4>{10}
```

e avere una presentazione che mostra ogni slide per dieci secondi, e al termine volta pagina (se la pagina successiva esiste).

Con questo abbiamo posto le basi per creare una presentazione autopresentante con Beamer. Siamo già in grado di farne una, a meno di suoni e animazioni, di cui parleremo nei prossimi paragrafi.

5 Suoni e filmati nei frame

Beamer ha la capacità di inserire file sonori e animati nelle presentazioni. A onor del vero, è grazie alle caratteristiche del formato PDF che una cosa del genere è possibile. In caso vogliate approfondire le caratteristiche del formato, potete fare riferimento a PDF. Vediamo nel dettaglio come inserire questi file multimediali nelle nostre presentazioni.

5.1 Suoni

Beamer fornisce il comando per inserire un suono in un frame. Questo comando, `\sound`, si trova nel pacchetto multimedia. Ci permette di far eseguire un suono all'apertura di una slide o alla pressione di un tasto. Questa funzionalità, abbinata alla possibilità di temporizzare la durata della persistenza delle pagine, ci permette di calibrare esattamente (o quasi) il tempo oltre il quale far cambiare pagina.

A causa di un bug presente in alcune versioni di Acrobat Reader, dobbiamo essere molto precisi nel passare i dati di codifica del brano da riprodurre.

Inoltre, sebbene sia in teoria possibile riprodurre sia brani in formati non compressi che compressi, il lettore di PDF di Acrobat non riesce a riprodurre questi ultimi, e gli unici due formati funzionanti sono `.aif` e `.au`. Dunque teniamo a portata di mano un programma per la conversione di formati sonori.

La mia esigenza era quella di eseguire un brano vocale (la mia voce che presentava il particolare della pagina) in ogni blocco in un frame. Dunque il codice aveva tanti `\sound` quanti erano i `block` in un frame. All'esecuzione, però, era sempre il primo file vocale del frame a essere riprodotto all'apparizione di un blocco. La soluzione più "comoda" è stata evitare i blocchi in ogni frame e includere un unico file sonoro all'inizio del codice del frame.

Il comando usato nella presentazione è `\sound[autostart,automute,bitspersample=8,inlinesound,channels=2]{}{slide22.au}`. Questo impone che il suono venga riprodotto all'apertura del frame, che la riproduzione termini alla fine del file sonoro, e che i dati del file sonoro siano quelli indicati.

5.2 Filmati

Anche l'inclusione di filmati è resa molto semplice da Beamer: il comando `\movie` è quello d'elezione per inserire video in formato AVI o MPEG. Non l'ho usato nella presentazione perché non era quella la mia esigenza. L'unica cosa da ricordare è che `\movie` è incluso nel pacchetto multimedia, distribuito con Beamer, ma come pacchetto separato, e utilizzabile anche indipendentemente da Beamer.

6 L'animazione di grafica vettoriale

6.1 Creazione dei fotogrammi

Ho avuto bisogno di inserire una piccola animazione in grafica vettoriale nella presentazione. Tale animazione è stata realizzata a partire da una serie di PDF prodotti con LATEX per mostrare enfaticamente come lavorava la prima versione del pacchetto combelow. Il mio scopo era rendere graficamente i passaggi del programma per spiegare più semplicemente l'algoritmo sottostante: ciò non vuol dire che l'animazione mostrasse *esattamente* quanto fatto dal pacchetto.

Non sono stati pochi i problemini da risolvere, sebbene tutti legati ai PDF da animare e non a LATEX e ai suoi pacchetti. Ne parlo qui a beneficio di quei lettori che potrebbero essere interessati a modificare un PDF in maniera diretta.

Tanto per non dimenticare, stiamo parlando del problema del posizionamento di un segno diacritico a forma di virgola (*comma below* o *comma accent*) sotto una lettera. La prima versione del pacchetto calcolava la posizione finale della virgola a partire dalla sua posizione canonica dopo la lettera, in base alla larghezza della lettera stessa. Dunque, le uniche cose che conosciamo per l'animazione sono

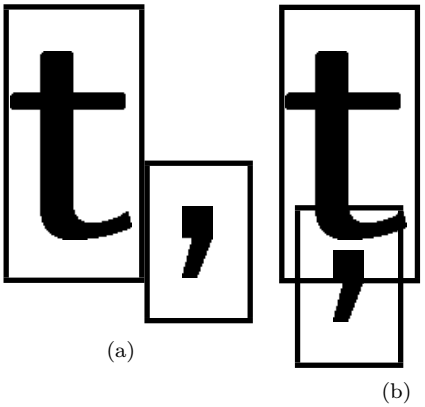

FIGURA 4: Configurazioni iniziale e finale per l'animazione di un'immagine in grafica vettoriale. Notate che la figura b è appositamente imbandierata a sinistra: è la virgola che si sposta verso la t, e non viceversa, o l'una verso l'altra vicendevolmente

le configurazioni iniziale e finale, mostrate nella figura 4.

Mentre la figura 4(a) è data dalla sequenza di testo `t,`, 4(b) è data dal comando `\cb{t}`, così come spiegato in PIGNALBERI (2009). Avrete notato che la figura 4(b) non è centrata; ciò fa pensare che il bordo destro non stia vicino all'estremità destra del disegno. Chiariremo tutto tra poco.

Generare una figura vettoriale come quella nella figura 4(a) è molto semplice: scriviamo un documento LATEX che contenga un codice come

```
\sffamily\textbf{
\Huge
\setbox0\hbox{t}
\setbox1\hbox{,}
\framebox[\wd0]{t}\framebox[\wd
    1]{,}}
```

senza numero di pagina o altro, compiliamolo con PdfLATEX per ottenere un PDF (che sarà in grafica vettoriale) e ritagliamo la figura con pdfcrop.

In accordo al pacchetto combelow v0.99a, la figura finale 4(b) può essere generata da un codice come questo:

```
\sffamily\textbf{
\Huge
\setbox0\hbox{t}
\setbox1\hbox{,}
\framebox[\wd0]{t}\raisebox{-.3
    ex}{\hskip-.85ex\framebox[\wd
    1]{,}}}
```

che ha un parametro numerico per `\hskip` in sostituzione dei calcoli del vero codice.

Generare le figure intermedie implica modificare i valori per `\hskip` e per `\raisebox`. Valori di `hskip` tra -0.1 e -0.85 avvicinano la virgola alla lettera fino a centrarla su di essa; valori di `\raisebox` tra -0.1 e -0.3 abbassano la virgola fin sotto la lettera. La tabella 1 riassume i valori

di posizionamento orizzontale e verticale per ogni fotogramma.

Come evidente dalla tabella 1, i fotogrammi non hanno le stesse dimensioni, né orizzontali, né verticali. Se provassimo ad animare una sequenza composta da quei fotogrammi otterremmo un pessimo risultato: ogni fotogramma verrebbe ridimensionato alle dimensioni del primo fotogramma. In questo caso vedremmo tutti i fotogrammi successivi al primo (che è il più largo e il più corto) deformarsi per allargamento e per accorciamento, e un'animazione che non renderebbe giustizia al vero processo.

La prima cosa da fare, prima di realizzare l'animazione vera e propria, è rendere tutti i fotogrammi delle stesse dimensioni. Per fare questo editeremo i file PDF come se fossero file di testo.

Per semplificare il lavoro, riporto qui le prime righe del primo fotogramma, omettendo quelle con caratteri binari incomprensibili all'uomo:

```
1  %PDF-1.4
2  %...
3  4 0 obj <<
4  /Length 38
5  /Filter /FlateDecode
6  >>
7  stream
8  ...
9  endstream
10 endobj
11 3 0 obj <<
12 /Type /Page
13 /Contents 4 0 R
14 /Resources 2 0 R
15 /MediaBox [0 0 20 25]
16 /Parent 5 0 R
17 >> endobj
```

La parte da modificare è quella relativa al bounding box dell'immagine, cioè quella della riga 15. Dovremo fare in modo che il bordo sinistro di tutti i fotogrammi rimanga fisso, cosicché non sembri la lettera a spostarsi, ma la virgola. Inoltre occorre fare in modo che anche il bordo superiore rimanga fisso, sempre affinché sia la virgola a muoversi anche verticalmente. Il bounding box (`/MediaBox` nel file PDF) è specificato da quattro numeri, che rappresentano la coordinata iniziale (primi due numeri, ascissa e ordinata dell'angolo in basso a sinistra) e finale (ultimi due numeri, ascissa e ordinata dell'angolo in alto a destra), e quindi la dimensione del box stesso. Nella tabella 2 sono elencati i valori dei *media box* calcolati da `pdfcrop`

Tabella 1: Valori di posizionamento orizzontale e verticale per i fotogrammi dell'animazione in grafica vettoriale

Numero fotogramma	**Shift orizzontale**	**Shift verticale**	**Fotogramma**
0	-0.0 ex	-.0 ex	
1	-0.1 ex	-.0 ex	
2	-0.2 ex	-.0 ex	
3	-0.3 ex	-.0 ex	
4	-0.4 ex	-.0 ex	
5	-0.5 ex	-.0 ex	
6	-0.6 ex	-.0 ex	
7	-0.7 ex	-.0 ex	
8	-0.8 ex	-.0 ex	
9	-0.85 ex	-.0 ex	
10	-0.85 ex	-.1 ex	
11	-0.85 ex	-.2 ex	
12	-0.85 ex	-.3 ex	

Tabella 2: Dimensioni del bounding box di ogni fotogramma dell'animazione di grafica vettoriale

Numero fotogramma	MediaBox originale	MediaBox finale
0	0 0 20 25	0 -4 20 25
1	0 0 19 25	0 -4 20 25
2	0 0 18 25	0 -4 20 25
3	0 0 17 25	0 -4 20 25
4	0 0 16 25	0 -4 20 25
5	0 0 15 25	0 -4 20 25
6	0 0 13 25	0 -4 20 25
7	0 0 12 25	0 -4 20 25
8	0 0 12 25	0 -4 20 25
9	0 0 12 25	0 -4 20 25
10	0 0 12 26	0 -3 20 26
11	0 0 12 28	0 -1 20 28
12	0 0 12 29	0 0 20 29

e quelli finali modificati a mano, in base alle dimensioni di tutti i fotogrammi. Questi numeri, che sembrano essere usciti come conigli dal cilindro di Mandrake, hanno invece un significato ben preciso. Il fotogramma più largo è il primo (quello numero 0), mentre il più alto è l'ultimo (numero 12). Dobbiamo fare in modo che tutti i fotogrammi siano larghi quanto il primo e alti quanto l'ultimo. Dobbiamo inoltre rispettare il vincolo di non far spostare a destra la lettera (quindi fissiamo il bordo sinistro del box, variando il destro, cioè lo spazio dopo la lettera, da dove la virgola parte) e di non farla spostare in basso (quindi fissiamo il bordo superiore e accettiamo di variare quello inferiore, cioè lo spazio sotto la lettera, dove la virgola arriva). Dopo il cambiamento delle dimensioni tutti i fotogrammi avranno i lati orizzontali di 20 pt, e quelli verticali di 29 pt.

Ora i fotogrammi sono pronti da animare e includere nel documento (naturalmente l'animazione non si vedrà nel documento cartaceo).

6.2 L'animazione

Ora che abbiamo lungamente visto come creare dei fotogrammi di un'animazione in grafica vettoriale, passiamo a vedere in pratica come ottenere l'animazione in un file PDF.

Beamer fornisce un comando utile a ottenere le animazioni di cui parliamo. Naturalmente non è necessario che i fotogrammi siano in grafica vettoriale: possono anche essere immagini raster (foto, illustrazioni o altro).

Beamer è in grado di creare due tipi di animazioni: il primo è relativo ai frame e ai relativi effetti di avvicendamento. I comandi a esso dedicati sono `\animate` e `\animatevalue` (trovate tutte le informazioni relative su beamer). Il secondo tipo di animazione riguarda invece l'avvicendamento di immagini esterne. Il comando dedicato è `\multiinclude` e si trova nel pacchetto xmpmulti. Io ho provato a usarlo, ma l'ho trovato criptico e scomodo da usare. Inoltre non sono riuscito a ottenere il risultato che mi aspettavo. Lo cito qui solo per completezza di informazione.

Un altro pacchetto che possiamo usare, quello che ho usato io per ottenere l'animazione di cui parlo, è animate. Questo pacchetto serve a ottenere animazioni a partire da una serie di fotogrammi di grafica vettoriale o immagini raster, o da grafica *inline*. Basato su JavaScript, consente animazioni molto versatili.

Appena incluso il pacchetto nel documento, abbiamo a disposizione i due comandi `\animategraphics` e `\animateinline`. Mentre il secondo serve a generare animazioni a partire da materiale composto al momento, il primo usa immagini preparate in precedenza. Visto che i fotogrammi erano già pronti, `\animategraphics` è proprio il comando che ho usato. La sua sintassi è `\animategraphics[<opzioni>]{<fotogrammi per secondo>}{<nome comune dei file>}{<primo>}{<ultimo>}`.

Nella fattispecie, nella presentazione ho usato `\animategraphics[autoplay,loop]{6}{tcomma-}{0}{12}`.

7 Conclusioni

Ho curato la realizzazione della presentazione qui studiata con TEXLive 2007. È possibile che le versioni successive abbiano superato, o lo faranno a breve, i pochi problemi esposti. Gli amanti di LATEX non ne saranno certo scoraggiati, e indirizzati verso programmi di presentazione alternativi.

Riferimenti bibliografici

beamer (2009). *The beamer class.*

PDF (2008). *Document management — Portable document format — Part 1: PDF 1.7.* Adobe Systems Incorporated, 1ª edizione.

PIGNALBERI, G. (2009). «combelow: abbasso i segni diacritici di serie b». In *ArsTEXnica*. GuIT, Pisa, 8, pp. 70–75.

▷ Gianluca Pignalberi
g dot pignalberi at alice dot it