Inserire Un Logo Con Avisynth

Scritta da aytin il 25-10-2006

CÃ pita che i loghi non vadano sempre rimossi ma a volte Ã¨ necessario inserire un proprio logo

---

Guida all'inserimento di un Logo con Avisynth

[Uso dei filtri di Virtualdub: Logo.vdf]

- SCRITTA DA : ***..::Aytin::..***
- COLLABORAZIONE, SUPPORTO E REVISIONE : ***..::DivXmania Staff::..***

Avisytnh oltre che a fare un sacco di cose simpatiche, come ogni buon tool grafico, è estendibile grazie ai plugin.
I cosidetti "filtri" non sono altro che dll che ne estendono le funzionalità già avanzate per conto proprio.

Oltre che i filtri canonici, costruiti specificatamente per Avisynth, è possibile importare filtri provenienti da altre fonti.
VirtualDub per es.

Facendo un rapido riassunto scopriamo che l'insieme dei filtri di Avisynth può essere alimentato da:

- dll specifiche del tool
- Filtri per VirtualDub (**.vdf*)
- VFAPI plugins (TMPGEnc)

Tutto questo si ottiene rispettivamente con:

- **loadPlugin**
- **loadVirtualDubPlugin**
- **LoadVFAPIPlugin**

Soffermeremo la nostra attenzione sulla 2_a primitiva: **loadVirtualDubPlugin**

Come si definisce uno script che vuole usare un filtro VirtualDub al suo interno?

Molto semplice.
Come in una dll normale, si importa e si usa.

Importazione:
loadVirtualDubPlugin

("*path****nome_filtro.vdf**", "*mio_nome_funzione*", *nRoll*))

Invocazione:

mio_nome_funzione(*par*$_1$, *par*$_2$,...*par*$_n$).

Anche se non sembra, è veramente tutto qua.
Gli inconvenienti nell'usare un filtro per VirtualDub stanno nel fatto che:

- Analogamente agli altri filtri, **occorre conoscere la segnatura del plugin** (vale a dire che bisogna sapere l'ordine e il tipo con il quale il plugin si aspetta che i parametri gli vengano passati) e non è sempre così banale.
  Spesso la documentazione è insufficiente. La form di configurazione presente su VirtualDub **non basta per stabilire in che ordine vadano invocati**.
- lo spazio-colore per i filtri VirtualDub è RGB32, Avisynth lavora preferibilmente su YV12. Ciò significa leggera perdita di informazioni se nello script, come può capitare, si passa da RGB32 a YV12 e viceversa.
  Oltre al fatto che l'uso di RGB32 rallenta la codifica (a meno che non abbiate un razzo al posto del PC).

Vediamo che accorgimenti si possono prendere per rendere l'importazione di un filtro per VirtualDub il più naturale possibile.

Definizione di LoadVirtualDubPlugin

loadVirtualDubPlugin("*path****nome_filtro.vdf**", "*mio_nome_funzione*", *preRoll*))

- Il 1° parametro è banalmente il path del filtro
- Il 2° è il nome della funzione con cui si mappa l'invocazione del filtro
- Il 3° parametro è un pò delicato. Rappresenta il numero di frames che tale filtro potrebbe aver bisogno di processare prima della sua applicazione. Il valore di default è 1. Calibrare male questo valore potrebbe provocare un crash dell'applicazione che fa uso di Avisynth

Dal momento che l'obiettivo della guida è di inserire un logo, facciamo un esempio pratico della teoria enunciata prima con il plugin **Logo.vdf**, il filtro per VirtualDub che permette di far comparire un logo, di regolarne fade, animazione, trasparenza ecc. e di cui, più avanti, verrà data una breve spiegazione dei parametri.

SOFTWARE NECESSARI

- Logo.vdf 1.5 o 1.7b2
- VirtualDub 1.6.16 o VirtualDub Mod 1.5.10.2
- Avisynth 2.56a o superiori

PREMESSA

Logo 1.5 è la versione stabile ma funziona bene con VirtualDub.
Con VirtualDub Mod capita che appaia il logo ma che il resto dello schermo rimanga nero.
Logo 1.7b2 è la versione consigliata, pur essendo una versione beta si dimostra molto efficace con entrambi.
E' essenziale per il buon funzionamento, che il logo sia un bitmap a 24 bit, altrimenti verrà segnalato un

errore.
Per il resto, occorrerà fare attenzione a nomi e path utilizzati, il resto delle guida dovrebbe essere sufficientemente esauriente.

Un grazie all'utente ZAPPATORCORTESE che, coerentemente al suo nick, si è dimostrato molto disponibile nel fornire feedback continui grazie ai quali ho potuto "correggere il tiro" su alcuni aspetti rimasti poco chiari.

1° APPROCCIO

QUOTE

```
LoadPlugin("C:\PROGRA~1\GORDIA~1\DGMPGDec\DGDecode.dll")
LoadPlugin("C:\PROGRA~1\GORDIA~1\AviSynthPlugins\ColorMatrix.dll")
LoadPlugin("C:\PROGRA~1\GORDIA~1\AviSynthPlugins\Undot.dll")

mpeg2source("C:\TEMP\Prove\prova.d2v", idct=0, info=3)
ColorMatrix(hints=true)
crop(2,14,714,548)
Undot()
BicubicResize(608,320,1/3,1/3)

ConvertToRGB32()
loadVirtualdubPlugin("c:\TEMP\prove\Logo.vdf", "myVD_Logo", 1)
myVD_Logo("C:\TEMP\prove\Logo.bmp",
 \ 0,0,128,1,0,0,255,0,0,0,0,0,40,120,80)
ConvertToYV12()
```

Questo script (le ultime 4 righe in particolare, solo quelle), sono sufficienti a far comparire il logo.
Distinguiamo in particolare:

- Conversioni di spazio-colore (1_a e 4_a riga)
- Importazione plugin di VirtualDub (2_a riga)
- Invocazione della funzione con i valori settati secondo quanto previsto dal plugin (3_a riga)

2° APPROCCIO

Per aumentare la leggibilità dello script si sposta l'invocazione del filtro in un altro script, che verrà importato. Ho scritto Logo.avsi in modo che gestisca anche la directory dei plugins di VirtualDub.

Script

QUOTE

```
LoadPlugin("C:\PROGRA~1\GORDIA~1\DGMPGDec\DGDecode.dll")
LoadPlugin("C:\PROGRA~1\GORDIA~1\AviSynthPlugins\ColorMatrix.dll")
LoadPlugin("C:\PROGRA~1\GORDIA~1\AviSynthPlugins\Undot.dll")
import ("C:\TEMP\prove\Logo.avsi")

mpeg2source("C:\TEMP\Prove\prova.d2v", idct=0, info=3)
ColorMatrix(hints=true)
crop(2,14,714,548)
Undot()
BicubicResize(608,320,1/3,1/3)

ConvertToRGB32()
VD_logo("C:\TEMP\prove\","C:\TEMP\prove\Logo.bmp",
\ 0,0,128,true,0,0,255,0,false,0,0,0,40,120,80)
ConvertToYV12()
```

Logo.avsi (presente nel mio caso in: **C:\TEMP\prove**)

QUOTE

```
function VD_Logo(clip clip, string "filter_path", string "filename",
 \ int "x", int "y", int "opacity", bool "transparent",
 \ int "xr", int "xg", int "xb", int "tolerance",
 \ bool "animate", int "start", int "duration", int "loops",
 \ int "fadeinlen", int "fadeoutend", int "fadeoutlen")
{
 # fissa la directory Plugins di default che sarà usata se non specificata nei parametri
 global VirtualDub_plugin_directory =
default(filter_path,"c:\programmi\VirtualDub\Plugins\")

 LoadVirtualdubPlugin(VirtualDub_plugin_directory+"Logo.vdf", "_VD_Logo", 1)
 return clip._VD_Logo(default(x,0), default(y,0), default(opacity,128),
 \ default(transparent,true)?1:0, default(xr,0), default(xg,0),
 \ default(xb,255),default(tolerance,0), default
 \ (filename,VirtualDub_plugin_directory+"logo.bmp"),
 \ default(animate,false)?1:0, default(start,0), default(duration,0),
 \ default(loops,0),default(fadeinlen,0),
 \ default(fadeoutend,0), default(fadeoutlen,0))
}
```

La funzione da importare non fa nulla di importante salvo che:

- invocare, come prima, il filtro restituendo il clip
- settare un default per la funzione in modo da poterla invocare anche in questa forma:
  VD_Logo(), che equivale a
  VD_Logo(0,0,128,1,0,0,255,0,"c:\programmi\VirtualDub\Plugins\",0,0,0,0,0,0)

I due approcci, pur essendo molto molto simili, sono differenti in un punto.
Nel primo devo trattare direttamente il plugin, nel secondo è stato virtualizzato.
In entrambi i casi, sull'importazione diretta o sulla costruzione dello script da importare, è stato necessario sapere **come** invocare il filtro.

N.B. Nello script ***Logo.avsi***, è fissata come directory di default per i plugin di VirtualDub **c:\programmi\VirtualDub\Plugins**.
Se i plugins si dovessero trovare in una cartella differente, basta modificare il default della variabile *globa* segnata in rosso.
Ricordo che questo script costituisce un template per tutti i plugins di VirtualDub che si vogliano mappare all'interno di Avisynth. La variazione riguarderà chiaramente la dichiarazione dei parametri e il loro settaggio nel default.

Dettagli dello script VD_LOGO

- **filter_path**: (*default* = "C:\programmi\VirtualDub\Plugins")
  Path del plugin *Logo.vdf*.
- **filename**: (*default* = "C:\programmi\VirtualDub\Plugins\logo.bmp")
  Path del logo.
- **x,y**: (*default* = (0,0) -> angolo in alto a sinistra)
  Coordinate iniziali su cui posizionare il logo.
- **opacity**: 0, 255 (*default* = 0)
  Regola la trasparenza di tutto il logo
- **transparent**: 0, 1 (*default* = 0)
  Abilita la trasparenza di **un colore** presente nel logo, fissato nella tripla che segue.
- **xr,xg,xb**: 0, 255 per tutti (*default* = (0,0,0))
  Codice RGB del colore
- **tolerance**: 0,255 (*default* = 0)
  Fissa il margine entro il quale un colore è considerato trasparente. In pratica permette di fissare un intorno dei coefficienti RGB. Tutti i pixel del logo aventi i codici RGB compresi in tale intorno, vengono resi trasparenti.
  Es. se transparent=true, RGB=(70,50,50) e tolerance 20, vuol dire che anche RGB=80,50,50 verrà reso trasparente ma anche RGB=38,38,38.
  In generale saranno resi trasparenti **tutti i colori che vanno da abs(RGB-*tolerance*) a RGB+*tolerance***. Molto utile quando la zona da rendere trasparente non è un colore uniforme.
- **animate**: 0, 1 (*default* = 0)
  Attiva l'animazione del logo se il filename è un nome del tipo: *nome_logo***0000**.bmp. Le altre bitmap verranno caricate automaticamente. L'importante è che ci siano i 4 zeri.

- **start**: 0, *maxNumeroFrames* (*default* = 0)
  Frame iniziale a partire dal quale viene mostrato il logo
- **duration**: 0, *maxNumeroFrames* (*default* = 0)
  Se l'animazione è disattivata, corrisponde al numero di frames in cui il logo viene mostrato.
  Se l'animazione è attivata, corrisponde al numero di frames in cui una bitmap viene mostrata prima che si passi alla successiva.
- **loops**: 0, *n* (*default* = 0)
  Solo per animazione attivata. E' il numero di volte in cui viene mostrata l'intera sequenza di bitmap.
- **fadeinlen**:0, *maxNumeroFrames* (*default* = 0)
  lunghezza del fade-in
- **fadeoutend**:0, *maxNumeroFrames* (*default* = 0)
  lunghezza del fade-out
- **fadeoutlen**:0, *maxNumeroFrames* (*default* = 0)
  numero di frames necessari per completare la dissolvenza.
  Es. se fadeinlen=100, fadeoutlen=300, fadeoutend=400 vuol dire che il logo diventa visibile dopo 100 frames dopodichè inizia gradualmente a scomparire fino a svanire del tutto dopo 400 frames.
  Chiaro no? Provare per credere.

ESEMPI:

QUOTE

```
VD_logo(filter_path="C:\TEMP\prove\", filename="C:\TEMP\prove\dm.bmp", opacity=128)
```

Mostra il logo con una trasparenza del 50%

QUOTE

```
VD_logo(filter_path="C:\TEMP\prove\", filename="C:\TEMP\prove\dm.bmp", x=100, y=100,
\ opacity=255, transparent=true, xr=51, xg=153, xb=102)
```

mostra il logo posizionato alle coordinate (100,100) con trasparenza per il colore verde dello sfondo.
N.B. Affinchè lo sfondo scompaia, è necessario che il suo colore sia quello corrispondente a xr,xg,xb.
Basta un semplice programma di fotoritocco per verificare.

Infine, senza immagini perchè avrebbe maggior senso un filmato, un esempio analogo a quello precedente ma con sfumatura in ingresso e in uscita.

QUOTE

```
VD_logo(filter_path="C:\TEMP\prove\",filename="C:\TEMP\prove\dm.bmp", opacity=255,
\ transparent=true, xr=51, xg=153, xb=102, fadeinlen=100, fadeoutlen=50, fadeoutend=400)
```

mostra il logo con trasparenza per lo sfondo, fade-in fino a 100 frames, visibilità completa per altri 250 per poi oscuramento rapido nel giro di 50 frames.

..::Aytin::..