EMBEDDED DATA IN LSCRIPT

Come molti si voi sapranno Lscript ha la capacità di gestire alcuni dati tramite degli "embedded data", ossia dati incastrati.
È possibile salvare veri e propri file all'interno di uno script semplicemente creando del codice binario formattato in maniera particolare.
E come è possibile creare questo codice binario formattato? Come faccio io programmatore a rappresentare un file con del codice binario?
La risposta a questa domanda è un semplice script dal nome "GenData.ls" (fornito nell'allegato), programmato da Bob Hood.
Questo generic script permette di selezionare il file e di inserire il suo codice binario frmattato in modalità append a un qualsiasi file di testo.

Glyph

Cosa è una Glyph?
Di questo particolare Object agent i manuali non fanno menzione, solo nei docs aggiuntivi dell'Lscript nel SDK è possibile studiarne il funzionamento.
Le Glyph sono utilizzate visualizzare immagini su di un requester, in effetti svolgono lo stesso compito di un control ctlimage, ma con qualche differenza che esamineremo in seguito.
Il loro funzionamento si basa esattamente sugli embedded data dell'LScript.
Dopo aver usato GenData e quindi dopo aver creato il codice binario necessario è possibile creare una Glyph.
Il costruttore per una glyph è il seguente:

```
myglyph = Glyph(MyImage);

@data MyImage 21900
255 216 255 224 000
…….
……..
@end
```

Questo costruttore può ricevere come argomento tre parametri, il secondo e il terzo sono opzionali e non verranno esaminati.
Dopo la creazione del Glyph Object Agent per poterlo inserire in un pannello è necessario richiamare il seguente metodo nella funzione di drawing del pannello stesso:

```
drawglyph(myglyph, pixel_iniziale_X, pixel_iniziale_Y);
```

Ma perché utilizzare una glyph e non un ctlimage?
Le differenze sono intuibili, e in particolare quella fondamentale è una.
Ctlimage richiede sempre nel suo costruttore come argomento il percorso per trovare il file dell'immagine e questo causa qualche problema nel caso lo script dovesse passare da un computer all'altro.
Nel secondo computer infatti al momento dell'esecuzione Lightwave andrebbe a ricercare il file all'interno dell'hard disk con esito negativo.
Con una glyph invece l'immagine verrà sempre visualizzata senza errori in quanto parte del codice dello stesso script.

Primitive Generator

Analizziamo adesso il funzionamento di “Primitive Generator”.
Come adesso potrete intuire le immagini sul pannello sono tutte delle glyph.

Fino ad adesso abbiamo studiato come da un file fisico è possibile creare il suo codice binario corrispettivo da inserire in un Lscript.
Ma come è possibile scrivere poi quei dati sul disco fisso?

Fornisco questo simpatico esempio:

```
@version 2.5
@warnings
@script generic
generic
{
   audiodata = File("click.wav","wb");
   audiodata.writeData(click);
   audiodata.close();
   reqbegin("Audio Test");
   c1 = ctlbutton("Press here",75,"click_button");
   reqpost();
   reqend();
   filedelete("click.wav");
}
click_button
{
   audio("click.wav");
}

@data click 400
082 073 070 070 110 001 000 000 087 065 086 069 102 109 116 032 016 000 000 000
001 000 001 000 017 043 000 000 017 043 000 000 001 000 008 000 100 097 116 097
074 001 000 000 128 129 148 176 000 000 255 255 000 000 255 255 000 000 255 232
000 060 255 252 000 100 255 156 000 148 255 072 000 224 232 008 088 224 104 012
136 224 100 096 044 255 136 064 092 220 180 108 056 152 136 144 056 112 124 152
088 128 108 196 180 100 036 196 184 112 204 220 188 172 076 080 192 136 008 176
180 048 128 192 136 092 168 180 108 112 144 124 128 136 176 140 104 188 168 096
080 192 184 088 140 176 188 104 108 192 184 100 100 192 156 072 112 204 128 112
136 156 136 076 132 176 140 104 140 184 124 108 152 152 136 108 132 152 116 156
116 148 132 112 152 172 096 100 196 168 080 108 164 160 076 116 188 116 104 164
172 120 076 148 168 124 092 164 156 116 140 140 144 124 124 136 188 108 112 148
144 120 148 136 168 092 040 255 000 000 255 064 216 255 255 044 255 255 056 144
128 000 024 144 012 156 220 140 200 212 144 124 128 096 156 116 072 120 176 100
100 212 156 080 140 156 084 112 136 188 108 040 108 255 188 204 000 020 255 136
255 000 112 255 044 255 000 220 200 200 232 000 140 120 255 220 000 144 120 248
080 060 120 076 204 096 180 112 132 196 104 172 080 164 140 100 068 128 212 100
100 092 108 208 156 136 064 108 176 152 160 108 092 140 136 140 172 092 120 136
168 148 144 112 132 164 140 112 144 116 120 148 168 120 096 140 168 172 140 084
148 188 160 088 112 132 164 140 100 152 112 168 152 000
@end
```

Questo generic script fornisce un'interfaccia nella quale è presente un solo pulsante.
Al momento della pressione di quest'ultimo si potrà udire il tipico suono "click".

Riassumendo i passaggi sono:
- creare un File object agent
- usare il suo metodo writeData, al quale verrà passato come parametro l'embedded data
- chiudere il file.

Dopo aver fatto queste operazioni avremo a disposizione un vero e proprio file fisico.

Primitive Generator non fa altro che questo.
Al momento della programmazione ho creato delle primitive base e ho inserito il loro codice binario corrispettivo all'interno del mio script.
Ogni volta che l'utente vuole creare una primitiva il plug-in si preoccupa di creare il file fisico e caricarlo in scena.
Niente di più.
Naturalmente nel codice sono presenti dei controlli in più relativi all'interfaccia.

Spero che questo breve documento vi sia utile, in caso di dubbi e domande non esitate a contattarmi.