```
MOVIE2.TAR.GZ (V2.0) - A simple movie player module
13/02/97

Movie Player V2.0
-----------------

Author:     Vince Diesi / SCEE

Date:    13-Feb-97



Description:
------------
A simple movie player module with the following features:

       1: Plays movies at 24 or 16 bit at any horizontal screen
resolution.

       2: Pre-processing stage to find the maximum VLC buffer size.

       3: Well defined interface and self contained module that can be
easily
             included into your program without too many problems.

       4: Efficient use of memory.

       5: Lots of error checking.



Usage:
------
To play a movie set up a StrInfo structure and call the PlayStream()
function. That's it! Refer to main.c for an example.



PlayStream() Parameters:
------------------------
The parameters used by PlayStream() are contained within a StrInfo
structure
(defined in movie.h). Each member is described below:

       1: strName      Stream file name.

       2: mode         Playback mode. Set to STR_MODE24 for 24Bit and
STR_MODE16
                                      for 16Bit playback.

       3: drawBorders  If set to STR_BORDERS_ON the display buffer VRAM
will be
                                      cleared before streaming. If 24Bit
playback has been
                                      selected the display buffer VRAM
is ALWAYS cleared
                                      to ensure that no 16Bit data is
displayed in 24Bit mode
                                      and vice versa!

       4: scrWidth     Any valid horizontal screen resolution. NOTE:
```

	Vertical resolution is always 240 for NTSC and 256 for PAL; Interlace mode is not supported!
5: X, Y	Stream position within display. If playback is 24Bit the X coordinate must be even.
6: width, height	Stream width and height.
7: endFrame	Stream end frame.
8: vlcBufSize	Size of each VLC buffer to be allocated (in bytes) including the space required for the runlevel header. If 0 is specified the maximum VLC buffer size is allocated (~128K each).
9: volume	ADPCM (XA Audio) volume (values 0 to 127). The same volume is used for both the left and right channels.

In addition to a StrInfo structure you must also provide PlayStream() with
a simple input handler to detect the user exit buttons. The input handler
must return 1 for exit, 0 otherwise.

NOTE: Although PlayStream() conducts many error checks it does assume that
the parameters you give it via StrInfo are valid!

PlayStream() Return Values:

PlayStream() plays the movie and blocks the caller until any of the following
conditions occur.

1: The stream has been played to the end successfully (PLAYSTR_END).
2: The user has exited the player using the controller (PLAYSTR_USER).
3: A fatal error occurred before or during playback (PLAYSTR_ERROR).

Upon regaining control your program should test the PlayStream() return value
and act appropriately. This is especially important if PLAYSTR_ERROR is
returned as this signifies a CD error.

Finding The VLC buffer Size:

Playing movies is an extremely memory intensive process. Eventhough decoding the image as a series of strips removes the need for any screen size image buffers, the VLC buffers required still eat lots of memory.

By defining FIND_VLCBUF within the makefile and compiling the module the maximum size of each VLC buffer can be found for a particular stream. This can then be passed to PlayStream() (via vlcBufSize) so only the correct amount of buffer memory is allocated.

This pre-processing phase works as follows:
The maximum possible VLC buffers are allocated (~128K each). The program then
plays the stream and stores the biggest runlevel size found (using DecDCTBufSize). This value, plus the run-level header size, represents the
size of the biggest VLC buffer required to play that particular stream.

The maximum VLC buffer required varies between each stream. Therefore, this
pre-process must be carried out for each stream.

When using the player in this mode if any frames are skipped a warning message is displayed since the skipped frames may produce a bigger runlevel than the maximum found.

For the same reason, the runlevel size is only reported if the stream is played completely (i.e. without the user exiting).

ADPCM Volume Control:

The player assumes the master volume has been set. The ADPCM volume is set
using the volume member of StrInfo The master volume or SERIAL_A attributes
are not altered in anyway. Only the ADPCM volume is faded out once the user
exits the player. The original ADPCM volume (before PlayStream() is called)
is restored on exit.

Adding the Movie Module to your Program:

Since the movie module is self contained and has a well defined interface
adding it to your program should (I hope) be a very easy task. The only area that may cause problems is memory allocation. movie.c uses its own memory allocation function using __heapbase. You may want to change StrInit()
function to use your own memory management functions, or even the libapi malloc and free functions =;-).

Error Checking:

The movie player module conducts a number of error checks during

initialisation and playback to prevent any serious (or fatal) problems.

During initialisation PlayStream() returns if the CdSearchFile() call is unsuccessful. In addition, it will retry starting the CD and obtaining the
first frame from the ring buffer. If after 5 retries it cannot start streaming
PlayStream() will return.

During playback PlayStream() will never continously remain in any loop, instead it will timeout instead and try and continue as normal. Two examples,
are when waiting for the next frame or for the MDEC decoding to finish.
If during playback a frame cannot be obtained from the ringbuffer PlayStream()
will continue as normal; decoding the previous frame. However, if
five frames pass without obtaining a new frame the player will exit.

When an error occurs the value PLAYSTR_ERROR is returned.

Detecting The End of a Stream:

The movie player detects the end of streams in three ways.
If the frame obtained from the ringbuffer:

1: Has a frameNo greater than the internal frame count.

2: Has a frameNo less than the internal frame count.

3: Has different/incorrect dimensions.

When creating your CD I recommend you order your stream data sequentially.
A small dummy stream with no XA audio is also advisable after the last stream.

Compile Options:

Both psymake and wmake makefiles have been provided. A number of variables
can be defined/undefined within the makefile which allow different modes
of compilation (see makefile for details).

If compiling with FINAL do not define FIND_VLCBUF. This code was tested
with libs 3.6 and 3.7.

Testing Mode:

If TESTING is defined some helpful debug information is displayed during playback. For instance, the number of frames skipped, the total amount of
time to process each frame etc.

This mode can be useful in determining whether the stream has been generated correctly. For instance, it will highlight if no room has been reserved for the audio sectors. If this is the case then frame skipping will occur.

The testing mode will also highlight if the VLC decode time is taking too
long for the required frame rate. This may occur if using the MDEC Version 3
option within MovConv. This option increases the quality of the compression,
however, the VLC decode time is increased. By using the testing mode you can see if the processing of frames is slower than reading them in from CD
(i.e. filling the ring buffer and causing frames to be skipped).

Possible Additions:

If I get some free time (highly unlikely) I intend to add the following:

1: Fade out to black when user exits the player.

2: Movie looping.

3: Interface for overlaying menus over streams.

4: Use the new VLC functions (libs3.7 onwards) which don't require the VLC decode table to be resident in memory at all times. This will save you
an extra 64K when not playing a movie.

If you find any bugs please forward them to Developer Support. With any luck
by the time you read this the program should be bug free=:-D.

Vince Diesi
SCEE