Transition Module Interface for DiVA VideoShop™

by Iván Cavero Belaúnde

About This Document

This document describes version 1 of DiVA VideoShop's plug-in transition module interface. Transitionss based on these specifications should contain 1 in the version field of the 'XsnI' resource.

Introduction

DiVA VideoShop transition effects are plug-in modules which allow DiVA VideoShop to combine the image data in two sequences of frames. Each of this modules communicates with VideoShop via the spec defined in this document, receiving parameters and image data and signaling errors. The actual manipulation is done by each specific module.

DiVA VideoShop transition effects are by their very nature dynamic; they change over time. This is in contrast with DiVA VideoShop filter effects, which can be both static and dynamic (for more information, see the document *Filter Module Interface for DiVA VideoShop*).

Transition modules are asked to combine the two frames of two sequences and are given information specifying how far into the sequences the operation is being performed. This information is generally used by the effect module to determine the "intensity" of the transition effect. For example, if the effect gets told we are 7 frames into a 10-frame sequence in a fade, it then performs a 70%-30% mix of the two images.

Transition modules are asked to operate on frames spaced out at specific intervals; this interval is determined by the smoothness control in the Apply Transition dialog. For a 3 second long transition with the smoothness control set to 10 fps, the transition will be asked to operate on 30 frames.

The resource structure for transition effects is described in the document *Writing Plug-Ins for DiVA VideoShop*. The code for the transition should in a 'Vxsn' resource of ID 0.

Record Structure and Equates

```
#define XSParameters            0
#define XSPrepare    1
#define XSStart                 2
#define XSProcessFrame          3
#define XSFinish                4

enum {
    XSErrReported                   =    0x00010000,
    XSErrUseParamText0              =    0x00020000,
    XSErrOutOfDiskSpace             =    0x00030000,
    XSErrOutOfMemory                =    0x00040000,
    XSErrXsitionFileCorrupted   =    0x00050000,
    XSErrBadSystemVersion           =    0x00060000,
```

```
    XSErrBadQDVersion           =   0x00080000,
    XSErrBadQTVersion           =   0x000A0000,
    XSErrBadVSVersion           =   0x000C0000,
};
```

```
typedef struct XsitionRecord {
    long                version;
    ProcPtr  abortProc;
    ProcPtr  progressProc;
    Handle  parameters;
    long                timeScale;
    long                timeStep;
    RGBColor            background;
    PixMapHandle        srcA;
    PixMapHandle        srcB;
    GWorldPtr           destImage;
}   XsitionParams, *XsitionParamPtr;

typedef struct XsnInfoRec {
    long                version;
    OSType media;
    long                flags;
    OSType xsnSignature;
    long                xsnVersion;
    OSType xsnManufacturer;
    OSType descRsrcType;
    short               descRsrcID;
/*  Reserved for future expansion       */
}   **XsitionInfo;

typedef long XsnErr;

typedef pascal XsnErr (*DiVAXsitionPtr)(short selector, XsitionParamPtr
    transitionRecord, long *data, short currentStep, short totalSteps);
```

Parameter Description

When calling the transition, VideoShop loads and locks the ‘Vxsn’ resource, which contains the code, and JSRs to offset zero in the resource, expecting there a routine that adheres to the calling conventions described by DiVAXsitionPtr.

The parameters passed to the transition code are as follows:

- *selector*: specifies which operation the transition is asked to perform (one of XsnParameters/ XsnStart/ XsnProcessFrame/ XsnFinish). These operations are called in a specific order from DiVA VideoShop, which is documented below under Calling Order.

- *transitionRecord*: a pointer to a structure of type XsitionRecord, as above, specifying parameters for the transition.

- *data*: A VAR parameter for the transition’s private usage; it could be used, for example, to hold a handle pointing to the transition’s global storage. When first called, the contents of *data are initialized to NIL.

- *currentStep, totalSteps*: These parameters specify the total number of frames the transition will have, as well as which frame the transition is currently at. Together, they specify the ‘intensity’ of the effect.

Parameter Block Description

- *version* - Holds the version of the interface this version of VideoShop adheres to. The current version is version 1.
- *abortProc* - Contains a pointer to afunction with the following calling conventions:

```
pascal Boolean TestAbort();
FUNCTION TestAbort: BOOLEAN;
```

  This procedure should be called at least once during FiltProcessFrame, and at best several times a second. If the function returns TRUE, the operation should be aborted.

- *progressProc* - A pointer to a procedure of type:

```
pascal void UpdateProgress (long done, long total);
```

  This procedure should be called to update the progress bar in the effect preview window during long operations inside FiltProcessFrame. The first parameter is the number of operations completed, and the second one is the total number of operations.

- *parameters* - If the transition has certain settings that it allows the user to set via a dialog box, it should store these settings in a handle, and store the handle in this field, and not deallocate the handle upon exit. This will allow VideoShop in the future to call transitions to perform identical mixing operations, such as by macro processing or high quality reprocessing of images. Additionally, the settings in this handle should not make assumptions about the size of the source image. If need to store size-dependent settings and you are passed the parameters handle by VideoShop, you should check that the size matches, and if not, remap the size-dependent settings to the new size. An example might be a transition that performs an zoom from a specified point in the image. If the coordinates of the point are saved in the settings dialog, the size of the image in which it was specified should be saved as well, and if the transition is asked to perform the same operation on images of different size, it should remap the point's coordinates to the new size.

- *timeScale* - This specified the time scale (in frames per second) of the totalSteps, currentStep, and (below) timeStep parameters. DiVA VideoShop 1.0 currently will only perform operations at a time scale of 30 frames per second. This might change in the future, and this is provided to allow the plug-in to make decisions on how to apply the transition based on the absolute length (in time) of the frame sequence.

- *timeStep* - This parameter specifies the number of frames by which VideoShop will step forward between succeeding calls to the transition. This parameter is determined by the setting of the smoothness control in the Apply Transition dialog box; for example, the middle setting of the smoothness corresponds to 10 frames per second, which corresponds to a timeStep value of 3.

- *background* - The background color for the movie. If the transition performs any spatial transformations that will leave space uncovered by image data, it should fill any space with this color.

- *srcA, srcB* - Hold two standard 32-bit QuickDraw PixMapHandles containing the two source images for operation by the transition. This pixelmaps can be assumed to be unpurgeable; they, however, should not be assumed to be locked, and thus the transition must call **LockPixels** before accessing it. The state of both these pixel maps and the destination image must be restored before returning control to VideoShop. These pixelmaps must also not be used as scratchpads of any sort; they should

be considered read only, for altering the images in any way could cause confusion with frame differenced movies. **srcA** contains the image from the sequence that is earlier in time, and **srcB** contains the image from the later sequence.

- *destImage* - Holds a standard 32-bit QuickDraw GWorldPtr containing the destination image buffer. As with srcImage, this GWorld's pixel map is unpurgeable and unlocked. The

destination image is passed as a graphics world for compatibility with graphics accelerators; since some QuickDraw operations might executa asynchronously, the transition might need to check if drawing operations are finished. The system call that allows this, **QDDone**, requires a GWorld as a parameters.

Calling Order

The user invokes the transition plug in by clicking on the Apply button in the VideoShop's Apply Transition dialog. VideoShop proceeds to load the plug-in resource into memory and calls it with the following sequence of values in the selector parameter:

• *XSParameters*

Asks the plug-in to put up a dialog box for user-settable parameters, and save them in a relocatable memory block whose handle is stored in the parameters field of xsnRecord. This field is initialized by VideoShop to nil before calling.

At this point in time, the plug-in can rely on the srcA and srcB fields in the parameter block to be valid, and containing the first images in the sequence of images that the transition is to be applied to. This is done to allow the plug in to offer "preview" functionality from the parameters dialog box.

This routine is always called by VideoShop, whether it already has settings for the transition or not. Thus it doubles up as an initialization routine, and the transition can check for specific hardware or software features and signal an error if necessary. If this routine returns an error, *XSFinish* will not be called.

• *XSPrepare*

If the transition is planning to allocate large buffers, this routine should attempt to check if the necessary memory for the buffers is available, whether in the VideoShop heap or in temporary memory, since by now VideoShop has performed most of the memory allocation in needs to apply the effect.

If you signal an error from this routine, VideoShop will abort the operation and *XSFinish* will not be called.

• *XSStart*

If the transition needs to allocate or initialize values, images, or tables needed for the entire length of the transition, it should do so here. Other preparations, such as initializing hardware for hardware-based transitions, could be performed here as well.

If you signal an error from this routine, VideoShop will abort the operation and *XSFinish* will not be called.

• *XSProcessFrame*

This is where all the work happens. When receiving this message, the transition is to process the entire frame. It is to call TestAbort at least once a second, as well as calling UpdateProgress as the frames are being composed. Please refer to the document *Writing Plug-Ins for DiVA VideoShop*, for detailed

information on calling TestAbort and UpdateProgress.

If you signal an error from this routine, VideoShop will abort the operation and *XSFinish* will be called to clean up whichever globals might have allocated.

• *XSFinish*

The transition should deallocate here all memory allocated by *XSStartSequence*. It should not, however, deallocate the settings handle stored by *XSParameters* in the **parameters** field for the xsnRecord structure. Deallocating that handle is the responsibility of VideoShop, and allows VideoShop to ask the transition to reprocess a sequence with the same parameters.

XsnInfo Structure Description

The XsnInfo structure is stored inside the 'XsnI' resource, and it contains the following information:

• *version* - The version of the calling interface this transition adheres to. This version is version 1.
• *media* - A descriptor for the type of media this transition operates on. Currently, VideoShop only supports transition that operate on video and graphics media, denoted by type '**vide**'. Future versions of VideoShop will include support for transition operating with other media, such as sound, and this information will allow your transition to remain compatible with those future versions.
• *flags* - 32 bits of flags for transitions to use. Currently no flags are defined.
• *xsnSignature* - A 4-character unique signature for the transition analogous to a creator type. This would allow future versions of VideoShop to uniquely identify each transition and be able to appropriately search for it to reapply it. We encourage developers to register the effect signatures with us; this will, in turn ease compatibility in the future as component manager-based effects specifications appear from Apple. Additionally, coupled with the *xsnVersion* and *xsnManufacturer* fields it would allow the parameter information for the effect (returned in **params->parameters**) to remain compatible across different versions of the effects modules.
• *xsnVersion* - A long word specifying the version of the transition module.
• *xsnManufacturer* - a 4-character unique signature for the transition manufacturer analogous to a creator type.
• descRsrcType, descRsrcID - The resource type and ID of a pascal string describing the transition. Currently unused

Most of the fields in the XsnInfo structure are currently unused. They are there to ease expansion of the transition functionality in the future into performing effects on other QuickTime media as QuickTime and VideoShop evolve, as well as expanding the capabilities of the transition modules. The transition you write WILL work with version 1.0 of DiVA VideoShop even if you ignore most of these fields (version and media are currently being used). If you do so, however, you do it at your own risk of incompatibility in the future.

Reporting Errors

VideoShop expects a longword as the error result from the transition module. If the operation was completed successfully, the transition should return 0 for the error. In the case a problem arises, VideoShop expects the transition to return a VideoShop-specific error code in the high word of the result, and the toolbox error (if any) in the low word. The predefined VideoShop error codes are as follows:

• *XsnErrReported* - VideoShop aborts the operation, but does not alert the user in any way. Your transition should only use this error code if it has reported the error itself or if the process is being halted as a result of TestAbort() returning true.

- *XsnErrUseParamText0* - VideoShop displays an error dialog box with the text from the first ParamText field. This is provided so that your transition can easily display custom error messages if it so desires.
- *XsnErrOutOfDiskSpace* - VideoShop displays the message “The effect could not be applied because there is not enough disk space in the recording bin.”
- *XsnErrOutOfMemory* - VideoShop displays the message “The effect could not be applied because there is not enough memory available.”
- *XsnErrXsitionFileCorrupted* - VideoShop displays the message “The effect module is corrupt. Please replace it with the original effect module.” This is provided so that if a resource you expected to find is not present you can exit gracefully and provide some feedback for the user. Should be useful for effects that save settings internally as well, since that can potentially corrupt the effect.
- *XsnErrBadSystemVersion* - VideoShop displays the message “This effect module will not function with the currently installed system software.”
- *XsnErrBadQDVersion* - VideoShop displays the message “This effect module will not work with the currently installed version of 32-bit QuickDraw.”
- *XsnErrBadQTVersion* - VideoShop displays the message “This effect module will not work with the currently installed version of QuickTime.”
- *XsnErrBadVSVersion* - VideoShop displays the message “This effect module will not work with this version of DiVA VideoShop.”

All other error codes will be reported to the user as “An unrecognized error has occurred while applying the effect.” When reporting the effect, if the low word of the error (the toolbox error) is nonzero, it will be displayed in parentheses immediately following the error description.