Technical Note TN2018

Importing animated GIFs

CONTENTS

This technote describes how to tell if a GIF file or dataref contains more than one frame, and why `GraphicsImportGetImageCount` will return a value of `1` even when a GIF is animated. It also introduces two new QuickTime 5 APIs - `CanQuickTimeOpenFile` and `CanQuickTimeOpenDataRef`, which are used to ask QuickTime if a file or dataref can be opened as a Movie or using a Graphics Importer.

Updated: [Apr 17 2001]

Overview

The GIF Graphics Importer will only display the first frame of animated GIF, so calling the GIF Graphics Importers `GraphicsImportGetImageCount` function will return a value of `1`.

Graphics Importers generally don't know about time, or how to combine key frames and difference frames. In addition, displaying the difference frames of an animation is generally not going to look very good without first drawing all the previous frames.

The Movie Toolbox, on the other hand, knows about time, and how to handle frame-differenced animations. Therefore, to draw the other frames in an animated GIF, import the GIF as a Movie.

To find out if a GIF file contains more than one frame, first open the GIF as a Movie, then count the frames in the video track. If there are multiple frames/samples treat it like a Movie; if there is only one frame/sample, the GIF is a single still image and the GIF Graphics Importer can be used instead.

`GetAnimatedGIFMovieFromFile` (see Listing 1) opens a file by using `NewMovieFromFile`, then counts the frames. It will return a valid inactive Movie if the opened GIF is animated, or `NULL` if it's a single image, in which case the caller can use a Graphics Importer.

Opening a GIF file as a Movie works faster if you don't pass the `newMovieActive` flag to `NewMovieFromFile`. If you decide to use the Movie after all (because it is an animated GIF), make the Movie active by calling `SetMovieActive`.

Listing 2 illustrates an alternate implementation of `GetAnimatedGIFMovieFromFile`. It creates a alias data reference then calls `GetAnimatedGIFMovieFromDataRef` (see Listing 3). Handling data references add the ability to open animated GIFs from memory and URLs.

Listing 3 shows how `GetAnimatedGIFMovieFromDataRef` performs the same task as listing 1 but using data references.

Listing 4 gives an example of how `CreateGIFHandleDataReference` creates a handle data reference with an added `'GIFf'` file type extension. If you have a GIF image in memory, this function can be used to easily create a handle data reference before calling `GetAnimatedGIFMovieFromDataRef`. Remember to dispose of the data reference handle when you're done with it.

Note:
In reality there's nothing specific to GIFs in the code listings except for the creation of the data reference in listing 4, they are named `GetAnimatedGIFMovie` only for the sake of this discussion.

Back to top

New for QuickTime 5

QuickTime 5 has added a couple of new APIs that can be used to determine whether a file or data reference can be opened as a Movie, using a Graphics Importer, or both. `CanQuickTimeOpenFile` determines whether the file, or a given file type, could be opened using a graphics importer or opened as a movie.

```
OSErr CanQuickTimeOpenFile(FSSpecPtr     fileSpec,
                           OSType        fileType,
                           OSType        fileNameExtension,
                           Boolean *     outCanOpenWithGraphicsImporter,
                           Boolean *     outCanOpenAsMovie,
                           Boolean *     outPreferGraphicsImporter,
                           UInt32        inFlags);
```

`CanQuickTimeOpenDataRef` is similar to `CanQuickTimeOpenFile` except that it uses a data reference instead of a file.

```
OSErr CanQuickTimeOpenDataRef(Handle      dataRef,
                              OSType      dataRefType,
                              Boolean *   outCanOpenWithGraphicsImporter,
                              Boolean *   outCanOpenAsMovie,
                              Boolean *   outPreferGraphicsImporter,
                              UInt32      inFlags);
```

Pass in `NULL` for parameters you don't particularly care about.

The `inFlags` parameter specifies flags that modify the Importer search behavior. They are:

- `kQTDontUseDataToFindImporter` - Don't use data in the file, speeds up the search but will cause QuickTime to report that it can not open files which aren't identified by a recognized file type or file name suffix.
- `kQTDontLookForMovieImporterIfGraphicsImporterFound` - Stop the search as soon as one way to open the file is found. Use this flag if you want to know whether a file can be opened with a graphics importer or as a movie, but you don't care which.
- `kQTAllowOpeningStillImagesAsMovies` - Consider opening still images as movies. When set, files that can be opened using a graphics importer will automatically be reported as being able to be opened as movies.
- `kQTAllowImportersThatWouldCreateNewFile` - Include importers which create new files. When clear includes only importers which can import in place without needing to create new files.
- `kQTAllowAggressiveImporters` - Set to include movie importers for file types like PICT and TEXT that aren't traditionally thought of as movies.

For more information regarding the new `CanQuickTimeOpenFile/DataRef` calls, see the QuickTime 5 Developer Delta Documentation.

```
// return a movie if the GIF is animated, NULL otherwise
Movie GetAnimatedGIFMovieFromAFile(const FSSpec *inFile)
{
MovietheMovie = NULL;
shorttheRefNum = 0;
longtheNumberOfSamples = 0;
OSErr  err = noErr;

err = OpenMovieFile(inFile, &theRefNum, fsRdPerm);
if (err) goto done;
err = NewMovieFromFile(&theMovie, theRefNum, NULL, NULL, 0, NULL);
CloseMovieFile(theRefNum);
if (err || NULL == theMovie) goto done;
theNumberOfSamples =
            GetMediaSampleCount(GetTrackMedia(GetMovieIndTrack(theMovie, 1)));

if (theNumberOfSamples == 1 ) {
DisposeMovie(theMovie);
theMovie = NULL;
}

done:
```

```
return theMovie;
}
```

Listing 1. GetAnimatedGIFMovieFromFile.

```
// return a movie if the GIF is animated, NULL otherwise
Movie GetAnimatedGIFMovieFromFile(const FSSpec *inFile)
{
Movie  theMovie = NULL;
AliasHandle theAlias = NULL;
OSErr  err = noErr;

err = QTNewAlias(inFile, &theAlias, true);
if (err) goto done;
theMovie = GetAnimatedGIFMovieFromDataRef((Handle)theAlias, rAliasType);
DisposeHandle((Handle)theAlias);

done:
return theMovie;
}
```

Listing 2. GetAnimatedGIFMovieFromFile (using an alias data reference).

```
// return a movie if the GIF is animated, NULL otherwise
Movie GetAnimatedGIFMovieFromDataRef(Handle inDataRef, OSType inDataRefType)
{
MovietheMovie = NULL;
longtheNumberOfSamples = 0;
OSErrerr = noErr;
if (NULL == inDataRef) goto done;
err = NewMovieFromDataRef(&theMovie, 0, NULL, inDataRef, inDataRefType);
if (err || NULL == theMovie ) goto done;

theNumberOfSamples =
            GetMediaSampleCount(GetTrackMedia(GetMovieIndTrack(theMovie, 1)));
if (theNumberOfSamples == 1 ) {
DisposeMovie(theMovie);
theMovie = NULL;
}
done:
return theMovie;
}
```

Listing 3. GetAnimatedGIFMovieFromDataRef.

```
// create a dataRef handle with a 'GIFf' file type extension
Handle CreateGIFHandleDataReference(Handle inData)
{
HandletheDataRef = NULL;
longtheFileTypeAtom[3] = {0};
OSErr  err = noErr;

// create a data reference handle for our data
err = PtrToHand(&inData, &theDataRef, sizeof(Handle));
if (err) goto done;
// no file name
err = PtrAndHand("\p", theDataRef, 1);
if (err) goto done;

// add the 'GIFf' 'ftyp' atom dataRef extension
theFileTypeAtom[0] = EndianU32_NtoB(sizeof(long) * 3);
theFileTypeAtom[1] = EndianU32_NtoB(kDataRefExtensionMacOSFileType);
theFileTypeAtom[2] = EndianU32_NtoB(kQTFileTypeGIF);

err = PtrAndHand(theFileTypeAtom, theDataRef, sizeof(long) * 3);

done:
if (theDataRef && err) {
DisposeHandle(theDataRef);
theDataRef = NULL;
}

return theDataRef;
}
```

Listing 4. `CreateGIFHandleDataReference`.

Back to top

Downloadables

	Acrobat version of this Note (48K)	Download

Back to top