Playing Audio Files with QTKit

Author: Mark Szymczyk
Last Update: December 5, 2007

This article shows developers how to use Apple's QTKit framework to play audio files in their Cocoa applications.

Introduction

QTKit is a framework that lets Cocoa programmers make use of QuickTime through Objective C classes instead of QuickTime's C API. By using QTKit you'll write less code than you will using the C API. If you doubt me, compare the amount of code in this article with the amount of code in my QuickTime audio article.

QuickTime is Apple's multimedia technology. While its main purpose is playing video, you can use QuickTime to display images and play audio. QuickTime can play multiple audio formats, including MP3, AAC (the format the iTunes Music Store uses), AIFF, and WAV files. With QTKit you can use the same set of code to play any audio file whose format QuickTime supports.

Apple introduced QTKit in Mac OS X 10.4. If you want to support earlier versions of Mac OS X, you must either use QuickTime's C API, which I covered in a previous article, or use the NSMovie class.

Setting Up Your Xcode Project

To use QTKit with Xcode, you must first create a Cocoa application project. Choose File > New Project to create an Xcode project. After creating the project, add the QTKit framework to your project. Select the Frameworks folder in the Groups and Files list and right-click (or control-click if you have a single button mouse) to open a contextual menu. Choose Add > Existing Frameworks from the contextual menu to add the QTKit framework.

Unless you're planning on having the user choose the files he or she wants to play, you'll be supplying the audio files for your application to play. In this case you should add the audio files you want to play to your Xcode project so they get added to the application bundle when Xcode builds the project. Select the Resources folder in the Groups and Files list and right-click to open a contextual menu. Choose Add > Existing Files from the contextual menu to add your audio files.

In your source code files that use QTKit you must include `QTKit.h`, the QTKit header file.

```
#import <QTKit/QTKit.h>
```

Reading the File

Reading an audio file with QTKit requires three steps. First, you must locate your application's bundle. Second, you must find the file in the bundle. Third, you must create a movie that you will use to play the audio.

Get the Bundle

Use the NSBundle class's `mainBundle` method to get your application's main bundle.

```
NSBundle* appBundle = [NSBundle mainBundle];
```

Find the File

Use NSBundle's `pathForResource` method to locate a file in the application bundle. This method takes two arguments. The first argument is the file name, and the second argument is the file's extension. Suppose you have a file named `MusicFile.mp3`. You can either supply `MusicFile.mp3` as the filename and `nil` as the extension, or you can supply `MusicFile` as the filename and `mp3` as the extension.

```
NSBundle* appBundle;
NSString* filename;
NSString* audioFile = [appBundle pathForResource:filename ofType:nil];
```

Create the Movie

Use the QTMovie class's `initWithFile` method to create a movie for the audio file you want to play. This method takes two arguments. The first argument is the path to the file you got when you called `pathForResource`. The second argument is an optional error argument that tells you what went wrong if the movie couldn't be created. If you don't care about the error, pass `nil`.

```
NSString* audioFile;
QTMovie* soundToPlay = [[QTMovie alloc] initWithFile:audioFile
      error:nil];
```

Playing a Movie Once

Playing a QTKit movie once is simple. Call the `play` method.

```
QTMovie* soundToPlay;
[soundToPlay play];
```

Playing a Movie Repeatedly

Games like to loop background music to avoid having to supply hours of audio for the game. Looping a movie is a little more complicated than playing it once. It requires two steps: telling QTKit to loop your movie, and using Cocoa's notification center to notify your program when the movie ends. When the movie ends you tell it to play again.

Telling QTKit to Loop Your Movie

To loop a movie you must tell QTKit to loop it by calling QTKit's `setAttribute` method. If you don't tell QTKit to loop the movie, you will notice a pause when you reach the end of the movie and play it again. You must set the attribute `QTMovieLoopsAttribute` to the value `YES`.

```
QTMovie* soundToPlay;
[soundToPlay setAttribute:[NSNumber numberWithBool:YES]
      forKey: @"QTMovieLoopsAttribute"];
```

Using the Notification Center

After telling QTKit to loop your movie, you must setup a notification that tells you when the movie stops playing. When you get the notification you tell the movie to play again.

To setup a notification first call the NSNotification class's `defaultCenter` method to get access to Cocoa's notification center.

```
NSNotificationCenter* notificationCenter =
      [NSNotificationCenter defaultCenter];
```

After you get the notification center, call NSNotification's `addObserver` method. It takes four arguments. The first argument is the observer, which is the object that gets sent the notification. Assuming you have an Objective C class for your sounds, the observer most likely will be `self`. The second argument is the selector, which is the message the notification center sends to the observer. The message is a method name. You supply the name of the method. The third argument is the name of the notification to look for. The notification `QTMovieDidEndNotification` tells you when the movie ends.

The final argument is the matching object. If you pass `nil` for the matching object, all notifications with the notification name you supply are sent to the observer. In the case of looping a movie, passing `nil` tells the notification center to send a message every time a movie ends. By passing a specific movie as the object, you're telling the notification center to send a message when that movie stops playing.

The following example tells the notification center to call a method called `playAgain` when reaching the end of a movie:

```
[notificationCenter addObserver:self
            selector:@selector(playAgain:)
            name:QTMovieDidEndNotification
            object:self];
```

The last task to loop movies is to write the method that will be called when a notification occurs. The name of the method must match the selector name you specified when calling `addObserver`. The method takes a pointer to NSNotification as its argument, but you don't need to use the argument to loop a movie. The following code shows an example of a `playAgain` method that tells a movie to play again:

```
- (void)playAgain:(NSNotification*)note
{
      [self play];
}
```

Stopping a Movie

Stopping a movie is simple. Call the `stop` method.

```
QTMovie* soundToPlay;
[soundToPlay stop];
```

When you call `stop`, the sound is paused. If you play the sound again while your application is running, the sound starts playing at the point where you stopped it.

Setting a Movie's Volume

Call the `setVolume` method to set the volume for a movie. Supply a floating-point number in the range 0.0 to 1.0, where 1.0 is maximum volume.

```
QTMovie* soundToPlay;
float desiredVolume;

[soundToPlay setVolume: desiredVolume];
```

QTKit has the `setMuted` method that lets you mute a movie. Pass the value `true` to mute the movie and `false` to unmute it.

```
- (void)mute
{
      [soundToPlay setMuted: true]
}

- (void)unmute
{
      [soundToPlay setMuted: false];
}
```

Conclusion

Included with this article is a simple application for you to download. It uses QTKit to loop an MP3 file. The application comes with full source code that you can use to play audio files in your Cocoa programs.