Speech

Cocoa > User Experience

2003-08-08

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Figures and Listings

Introduction to Speech

The Cocoa programmatic interface for speech enables an application to “pronounce” a string of text using a synthetic voice. In addition, an application can use this interface to “listen” for commands spoken by a user and act upon those commands. An application can combine both capabilities for a interactive user experience that enables the user to accomplish a multi-step task without depending on the keyboard or mouse. These capabilities not only benefit users of your application who have attention, vision, or other physical disabilities, but can be used by the application to convey or obtain critical information without forcing users to shift focus from what they’re doing.

The NSSpeechSynthesizer and NSSpeechRecognizer classes provide the Cocoa interface to the lower-level Carbon technologies of Speech Synthesis and Speech Recognition, respectively. If you require greater control of speech than permitted by the Cocoa classes, you may use the underlying Carbon frameworks instead.

Important: The NSSpeechSynthesizer and NSSpeechRecognizer classes are available in Mac OS X version 10.3 and later.

Organization of This Document

Speech in Cocoa consists of two general procedures:

- "Synthesizing Speech" (page 9) describes how to use an NSSpeechSynthesizer object to speak a string of text to users in a specific voice.
- "Recognizing Speech" (page 11) describes how to use an NSSpeechRecognizer object to apprehend spoken commands which your application can then act upon.

Synthesizing Speech

By using an NSSpeechSynthesizer object, you can make your application speak a word, phrase, or sentence to the user. This synthesized speech is an essential aid to those users with attention or vision disabilities. It is also useful when you want to draw the user's attention to something important when he or she might be distracted by something else on the screen.

Using an NSSpeechSynthesizer object to "pronounce" text is easy. You initialize the object with a voice and send a `startSpeakingString:` message to it, passing in an NSString object representing the text to speak. Optionally, you can implement one of several delegation methods either to accompany the words pronounced in some interactive fashion or to do something application-specific when speaking concludes.

Important: The NSSpeechSynthesizer class is available in Mac OS X version 10.3 and later.

Voices and Initialization

The essential attribute of an NSSpeechSynthesizer object is a voice. Speech Synthesis defines a number of voices for Mac OS X, each with its own recognizable speech characteristics (such as gender and age). You can view the list of system voices, and set the default voice, in the Default Voice pane of the Speech system preferences.

If you initialize an NSSpeechSynthesizer instance using the `init` method, the default voice is used. If for some reason you want another voice, initialize the instance with the NSSpeechSynthesizer method `initWithVoice:`. You can change the voice at any time with the `setVoice:` method.

By invoking the class methods `availableVoices` and `defaultVoice`, you can get the list of system voices and the current default voice, respectively. Each voice has multiple attributes, including name, age, gender, and language. By invoking the class method `attributesForVoice:`, you can get an NSDictionary object for a specific voice which contains these attributes. (The argument of this method is a voice identifier, a string of the form `com.apple.speech.synthesis.voice.`*`voiceName`*.) See the reference documentation for NSSpeechSynthesizer for the valid dictionary keys.

Speaking Text

Once you have initialized the NSSpeechSynthesizer object, send the `startSpeakingString:` message to it, passing it the text to speak. Listing 1 illustrates initializing the object with the default voice and then, prior to speaking a string fetched from a text field (`phraseField`), setting the voice as requested by the user in a pop-up list (`voiceList`).

Listing 1 Using an NSSpeechSynthesizer object

```
- (id)init {
```

```
    self = [super init];
    if (self) {
    synth = [[NSSpeechSynthesizer alloc] init]; //start with default voice
                                                //synth is an ivar
    [synth setDelegate:self];
    }
    return self;
}

- (IBAction)speak:(id)sender
{
    NSString *text = [phraseField stringValue];
    NSString *voiceID =[[NSSpeechSynthesizer availableVoices]
objectAtIndex:[voiceList indexOfSelectedItem]];
    [synth setVoice:voiceID];
    [synth startSpeakingString:text];
}
```

Note that this code example sets a delegate for the NSSpeechSynthesizer object in the `init` method. NSSpeechSynthesizer defines three methods to allow its delegate to become involved during the speaking of a string of text and after the speaking of a string:

- `speechSynthesizer:willSpeakWord:ofString:`, invoked before each word of the string is spoken, allows the delegate (for example) to visually highlight each word as it is spoken.
- `speechSynthesizer:WillSpeakPhoneme:`, invoked before each phoneme is pronounced, allows the delegate (for example) to animate a mouth pronouncing the phoneme.
- `speechSynthesizer:didFinishSpeaking:` is invoked when the speaking of the string ends. The second parameter of this method indicates whether the text was entirely spoken or was disrupted (as might happen if the user dismisses a spoken alert).

  You might implement this method to adjust the user interface appropriately when speaking ceases. Listing 2 exemplifies one such implementation.

Listing 2 An implementation of speechSynthesizer:didFinishSpeaking:

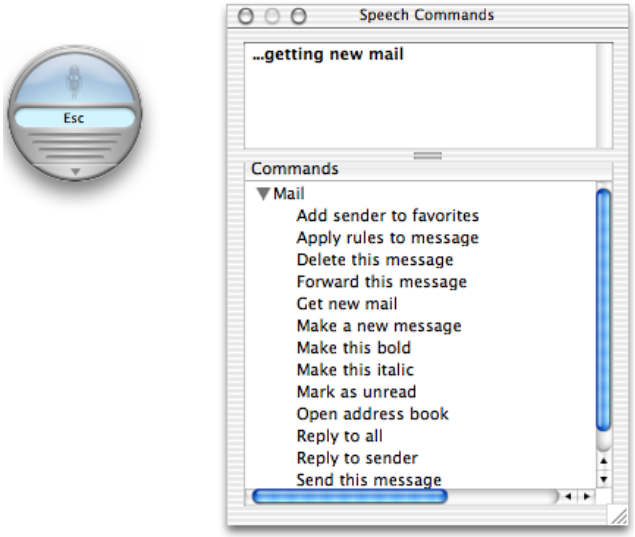
```
- (void)speechSynthesizer:(NSSpeechSynthesizer *)sender
didFinishSpeaking:(BOOL)finishedSpeaking
{
    [_textToSpeechExampleTextView setSelectedRange:_orgSelectionRange]; // Set
 selection length to zero.
    [_textToSpeechExampleSpeakButton setTitle:@"Start Speaking"];
    [_saveButton setTitle:@"Save As File..."];
    [_textToSpeechExampleSpeakButton setEnabled:YES];
    [_saveButton setEnabled:YES];
    [_voicePop setEnabled:YES];
}
```

Recognizing Speech

With an audio input device (such as a microphone) and an NSSpeechRecognizer object an application can listen for spoken commands and act upon those commands. Speech recognition is an essential aid for users with physical disabilities that limit their use of the keyboard and mouse. It can also be a convenience for all users by enabling them to control an application without forcing them to divert attention from what they're currently working on.

The centralized system service Speech Recognition is activated on a system whenever an application (including those listed in the Speech Recognition pane of System Preferences) begins listening through any Speech Recognition API, including those of NSSpeechRecognizer. When speech recognition is activated, an on-screen microphone and (optionally) the Speech Commands window appear. The Speech Commands window lists the current commands that can be given as well as acknowledgements from applications that have responded to recent commands. Figure 1 (page 11) shows what the microphone and Speech Commands window look like (in the context of the Mail application).

Figure 1 Screen microphone and Speech Commands window



Integrating speech recognition into a Cocoa application is simple. The important steps involve specifying the commands to listen for and then listening and responding to those commands. The remainder of this article goes into each of these steps in detail.

Preparing an NSSpeechRecognizer Object

To prepare an NSSpeechRecognizer for use, you must:

1. Allocate and initialize an instance of NSSpeechRecognizer.
2. Set the commands that the object should listen for using the `setCommands:` method.
3. Set a delegate for the NSSpeechRecognizer object.

Listing 1 shows how you might initialize an NSSpeechRecognizer object.

Listing 1 Preparing an NSSpeechRecognizer object

```
- (id)init {
    self = [super init];
    if (self) {
       NSArray *cmds = [NSArray arrayWithObjects:@"Sing", @"Jump", @"Roll over",
 nil];
        recog = [[NSSpeechRecognizer alloc] init]; // recog is an ivar
        [recog setCommands:cmds];
        [recog setDelegate:self];
    }
    return self;
}
```

Commands are words or short phrases, encapsulated as NSString objects, that are specific to the application. The recommended phrase length is three to six words. If your application has many commands, you can use the `setDisplayedCommandsTitle:` method to group them in the Speech Commands window under subheadings.

Listening For and Responding To Commands

Before your application can process spoken commands, you must activate the speech-recognition engine by sending a `startListening` message to the NSspeechRecognizer object. The engine then attempts to discern commands in the stream of words and phrases the user speaks into the microphone. If it identifies a command, the NSSpeechRecognizer object invokes the `speechRecognizer:didRecognizeCommand:` delegation method, passing in the command in the second parameter. To suspend the speech-recognition engine, send the NSSpeechRecognizer object a `stopListening` message.

You can instantaneously update the list of commands for which the NSSpeechRecognizer object listens by sending it a `setCommands:` message. Command updating occurs even when the object is actively listening.

The delegate should implement the `speechRecognizer:didRecognizeCommand:` delegation method to respond to each spoken command. Listing 2 shows an example implementation of this method. It also shows an action method that toggles between starting and stopping the recognition engine.

Listing 2 Starting the recognition engine and responding to commands

```
- (IBAction)listen:(id)sender
{
    if ([sender state] == NSOnState) { // listen
    [recog startListening];
    } else {
    [recog stopListening];
    }
```

```
}

- (void)speechRecognizer:(NSSpeechRecognizer *)sender didRecognizeCommand:(id)aCmd
 {

    if ([(NSString *)aCmd isEqualToString:@"Sing"]) {
        NSSound *snd = [[NSSound alloc] initWithContentsOfFile:[[NSBundle
mainBundle] pathForResource:@"HappyBirthday" ofType:@"aif"] byReference:NO];
        [snd play];
        [snd release];
        return;
    }

    if ([(NSString *)aCmd isEqualToString:@"Jump"]) {
        NSRect frm = [[phraseField window] frame];
        [[phraseField window] setFrameOrigin:NSMakePoint(frm.origin.x+20,
frm.origin.y+20)];
        return;
    }

    if ([(NSString *)aCmd isEqualToString:@"Roll over"]) {
        // .... some response here...
    }
}
```

Document Revision History

This table describes the changes to *Speech*.

Date	Notes
2003-08-08	First version of Speech.