

[articles](#)[Q&A](#)[forums](#)[features](#)[lounge](#)[?](#)

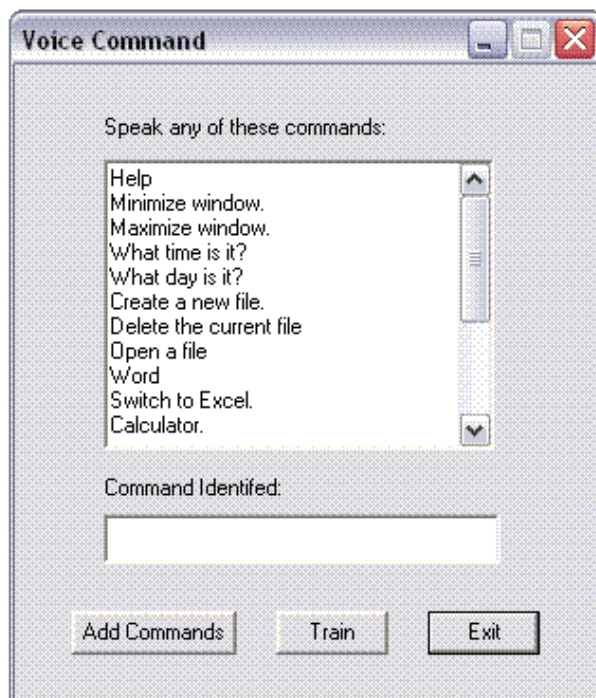
Voice Command

biju_786

26 Sep 2005

Rate me: ★★★★★ 4.71/5 (19 votes)

An article on the Voice Command of speech recognition.

[Download source - 14 Kb](#)[Download demo - 33.4 Kb](#)

Introduction

The Voice Command Demo demonstrates a simple speech recognition by showing you the commands it recognizes.

A speech recognition engine should be installed to run the program. You can download the Microsoft Speech Recognition Engine from [here](#).

The Voice Command interface is the high-level interface for speech recognition. It is designed to provide command and control speech recognition for applications. With this interface, a user gives the computer simple commands, such as "Open the file", and can answer simple yes/no questions. Command and Control does not allow speech dictation.

The Voice Command design mimics a Windows menu in behavior, providing a "menu" of commands that users can speak. Basically, to use voice commands, an application designs a Voice menu that corresponds to a window or state within the application. Most programs will have one Voice menu for the main window and one for every dialog box. Contained within every Voice menu is a list of voice commands that users can say. When they say one, the application is notified which command was spoken. "Open a file" and "Send mail to <e-mail name>" are typical voice commands. Each voice command has information in addition to the spoken command, such as a description string and a command ID.

Voice commands allow the user to control an application by speaking commands through an audio input device rather than by using the mouse or keyboard, giving the user hands-free control of the application. Voice commands involve the use of an audio input device, such as a microphone or a telephone, a speech recognition engine, and a Voice menu. When the user speaks a command into the audio input device, the speech recognition engine attempts to transcribe the spoken input into text. If the engine succeeds, it compares the command text to that of the commands in the active Voice menus. (A Voice menu contains a list of commands to which an application can respond.) If the engine finds a matching command in a Voice menu, it notifies the application of the match, and the application carries out the command.

Why Use Command and Control?

In general, use Command and Control recognition when:

1. It makes the application easier to use.
2. It makes features in the application easier to get to.
3. It makes the application more fun/realistic.

If an application uses speech recognition solely to impress people, it will work well for demos but will not be used by real users.

This sample program identifies a command spoken by the user from a set of commands, and displays it.

Requirements

Microphone

The user can choose between two kinds of microphone: either a close-talk or headset microphone that is held close to the mouth, or a medium-distance microphone that rests on the computer 30 to 60 centimeters away from the speaker. A headset microphone is needed for noisy environments.

Speech-recognition engine

Speech-recognition software must be installed on the user's system. Many new audio-enabled computers and sound cards are bundled with speech-recognition engines. As an alternative, many engine vendors offer retail packages for speech recognition or text-to-speech, and some license copies of their engines. If you don't have one, you can download one from [here](#).

Limitations

Currently, even the most sophisticated speech-recognition engines have limitations that affect what they can recognize and how accurate the recognition will be. This may seem like an impenetrable list, but a savvy application can design around these limitations.

Using the code

SpeechReg.cpp is the implementation file for the program.

1. Initialize the application

To use voice commands, you need to create a Voice Command object, register your application with the object, and then create a Voice Menu object to manage your application's voice menus. You create a Voice Command object by calling the **CoCreateInstance** function with the **CLSID_VCmd** class identifier and the **IID_IVoiceCmd** interface identifier. You must create a separate Voice Command object for each site that your application needs to use.

CoCreateInstance returns a pointer to the **IVoiceCmd** interface for the Voice Command object. Before it can perform other Voice Command tasks, an application must register itself by calling the **IVoiceCmd::Register** member function. Register specifies the site that the object represents and passes the address of the application's Voice Command notification interface to the Voice Command object.

After creating a Voice Command object and registering the application, you can use the **IVoiceCmd::MenuCreate** member function to open a voice menu and create a Voice Menu

object to represent the menu. **MenuCreate** retrieves the address of the **IVCmdMenu** interface for the Voice Menu object. You can use the interface's member functions to manage the menu and its commands.

The following example shows how to create a Voice Command object, register an application, and create a Voice Menu object. The example creates a temporary Voice Menu object; that is, the object is not added to the Voice Menu database maintained by the Voice Command object.

The function initializes OLE, creates an instance of the Voice Command object, registers the application with the object, and creates a temporary Voice Menu object. It returns **TRUE** if successful or **FALSE** otherwise. The function uses the following global variables and constants:

- **gpIVoiceCommmand** -- address of the **IVoiceCmd** interface for the Voice Command object.
- **gpIVCmdDialogs** -- address of the **IVCmdDialogs** interface for the Voice Command object.
- **gpIVCmdMenu** -- address of the **IVCmdMenu** interface for the Voice Menu object.

The **BeginOLE()** function begins the OLE and creates the Voice Command object, registers with it, and creates a temporary menu.

Shrink ▲ 

```

BOOL BeginOLE()
{
    HRESULT hRes;
    VCMDNAME VcmdName;
    LANGUAGE Language;
    PCIVCmdNotifySink gpVCmdNotifySink = NULL;
    PIVCMDATTRIBUTES pIVCmdAttributes;
    SetMessageQueue(96);

    CoInitialize(NULL);

    // Create the voice commands object
    hRes=CoCreateInstance(CLSID_VCmd, NULL, CLSCTX_LOCAL_SERVER,
                        IID_IVoiceCmd, (LPVOID *)&gpIVoiceCommand);

    // Get the dialogs interface pointer...
    hRes = gpIVoiceCommand->QueryInterface( IID_IVCmdDialogs,
                                           (LPVOID FAR *)&gpIVCmdDialogs );

    // Get the attributes interface pointer...
    // hRes = gpIVoiceCommand->QueryInterface( IID_IVCmdAttributes,
                                           (LPVOID FAR *)&gpIVCmdAttr );

    // Create/Register VCmd notification sink...
    gpVCmdNotifySink = new CIVCmdNotifySink;

    hRes = gpIVoiceCommand->Register( "", gpVCmdNotifySink,

```

```

        IID_IVCmdNotifySink, VCMDRF_ALLMESSAGES, NULL );

    if(FAILED(hRes))
        MessageBox(m_hwnd, "Error in registering", "Speech Reg", MB_OK);

    //The following code checks for a navigator app and
    //checks the state of voice commands

    hRes = gpIVoiceCommand->QueryInterface(IID_IVCmdAttributes,
        (LPVOID FAR *)&pIVCmdAttributes);
    if (pIVCmdAttributes)
    {
        pIVCmdAttributes->EnabledSet( TRUE );
        pIVCmdAttributes->AwakeStateSet( TRUE );

        pIVCmdAttributes->Release();
    };

    // Initialize command menu set variables...
    lstrcpy(VcmdName.szApplication, "Speech Reg");
    lstrcpy(VcmdName.szState, "Main");
    Language.LanguageID = LANG_ENGLISH;
    lstrcpy (Language.szDialect, "US English");

    // Create an empty command menu set...
    hRes = gpIVoiceCommand->MenuCreate( &VcmdName, &Language,
        VCMDMC_CREATE_TEMP, &gpIVCmdMenu );
    if( FAILED(hRes) )
        MessageBox(m_hwnd, "Failed to create a voice "
            "command set with MenuCreate()",
            "Speech Reg", MB_OK);

    return TRUE;
}

```

2. Adding Commands to a Voice Menu

After you create a Voice Menu object, you can add commands to the menu by filling an array of **VCMDCOMMAND** structures, copying the address and size of the array into an **SDATA** structure, and passing the address of the **SDATA** structure to the **IVCmdMenu::Add** member function.

The example in this section shows how to add a new set of commands to a Voice Menu object. The example consists of three functions: **UseCommands**, **GetCommands**, and **NextCommand**:

The **UseCommands** function deactivates the Voice menu, replaces any existing commands in the menu with a new set, and reactivates the menu. One of the parameters to **UseCommands** is the address of a buffer containing the list of command strings to enter. **UseCommands** passes the address of the command-string buffer to the **GetCommands** function, along with the address of an **SDATA** structure.

The **GetCommands** function converts the buffer to an array of **VCMDCOMMAND** structures and copies the address and size of the array into the **SDATA** structure.

The **NextCommand** function is a helper routine that **GetCommands** uses to retrieve individual command strings from the command buffer passed to **UseCommands**.

3. Responding to Voice Command Notifications

The Voice Command object calls an application's **IVCmdNotifySink** interface to inform the application of Voice Command events so that the application can respond to them. To receive notifications, the application must create a COM object that supports the **IVCmdNotifySink** interface and must pass the address of the interface to the Voice Command object when calling the **IVoiceCmd::Register** function.

The **IVCmdNotifySink** interface consists of a set of member functions that correspond to Voice Command events. When an event occurs on the site that the application is using, the Voice Command object calls the member function that corresponds to the event.

The following example shows how to define an object class that implements the **IVCmdNotifySink** interface:

```
class CIVCmdNotifySink : public IVCmdNotifySink {
private:
    DWORD    m_dwMsgCnt;
    HWND     m_hWnd;

public:
    CIVCmdNotifySink(void);
    ~CIVCmdNotifySink(void);    // IUnknown members
    STDMETHODCALLTYPE QueryInterface (REFIID, LPVOID FAR *);
    STDMETHODCALLTYPE AddRef(void);
    STDMETHODCALLTYPE Release(void);

    // IVCmdNotifySink members
    STDMETHODCALLTYPE CommandRecognize (DWORD, PVCMDNAME, DWORD,
                                         DWORD, PVOID, DWORD, PSTR, PSTR);
    STDMETHODCALLTYPE CommandOther    (PVCMDNAME, PSTR);
    STDMETHODCALLTYPE MenuActivate    (PVCMDNAME, BOOL);
    STDMETHODCALLTYPE UtteranceBegin  (void);
    STDMETHODCALLTYPE UtteranceEnd    (void);
    STDMETHODCALLTYPE CommandStart    (void);
    STDMETHODCALLTYPE VUMeter         (WORD);
    STDMETHODCALLTYPE AttribChanged   (DWORD);
    STDMETHODCALLTYPE Interference    (DWORD);};
typedef CIVCmdNotifySink * PCIVCmdNotifySink;
```

License

This article has no explicit license attached to it but may contain usage terms in the article text or the download files themselves. If in doubt please contact the author via the discussion board below.

A list of licenses authors might use can be found [here](#)

Written By

biju_786

Web Developer

 India

This member has not yet provided a Biography. Assume it's interesting and varied, and probably something to do with programming.

Comments and Discussions

You must [Sign In](#) to use this message board.

[First](#) [Prev](#) [Next](#)

Problem

Member 9960226 27-Apr-16 2:08

error

abhay9450135 20-Apr-13 23:07

syntax errors

veillee ahmed 1-Jul-12 15:05

Help

veillee ahmed 30-Jun-12 4:38

voice command

gracey115 17-Feb-12 5:22

voice command

gracey115 10-Jan-12 21:34

Facing error message after successful build operation

TRUSHAL RANE 21-Dec-11 18:30

Speech Recognition

2918119 5-Mar-09 1:42

when i try to build into VC2005 this project... 

Alex21stud 14-Jan-09 6:55

i don't know why it returns all failed 

wangyangtao 5-May-08 20:47

how integrate the my own speech recognition engine to sapi 5.1 

Mohan.M 25-Feb-08 22:35

Edit the layout 

Jasper91 16-Jul-07 7:30

Error, cannot open include file 'afxres.h' 

Jasper91 15-Jul-07 6:09

Re: Error, cannot open include file 'afxres.h' 

Christian Graus 15-Jul-07 6:13

Re: Error, cannot open include file 'afxres.h' 

Jasper91 15-Jul-07 7:38

Re: Error, cannot open include file 'afxres.h' 

Jasper91 15-Jul-07 9:49

Compatibility 

Piconze 2-Apr-07 4:44

Can SAPI Recongize Chinese&#65292; And how&#65311; 

early4932 18-Mar-07 19:51

MFC application 

mdahsan 20-Sep-06 10:56

Compatibility 

Kaos_rules 27-Jun-06 18:47

Need a program like this! 

geo_soft 20-Jun-06 0:22

one more thing 

usman2ba 13-May-06 13:22

add grammar 

TassadaqHussain 23-Oct-05 10:07

not SDK 5.1 

Gilbert Chauvaux 6-Oct-05 20:06

[Refresh](#)

1 2 Next 

 General  News  Suggestion  Question  Bug  Answer  Joke  Praise  Rant 
Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#)
[Advertise](#)
[Privacy](#)
[Cookies](#)
[Terms of Use](#)

Layout: [fixed](#) | [fluid](#)

Article Copyright 2005 by biju_786
Everything else Copyright © [CodeProject](#),
1999-2023

Web02 2.8:2023-05-13:1