Chapter 7

Microsoft® DirectX™ 3 Software Development Kit

DirectSetup

DirectSetup

C H A P T E R 7

About DirectSetup

DirectSetup is a simple application programming interface (API) that provides you with a one-call installation for the DirectX™ 3 components. This is much more than a convenience; DirectX 3 is a complex product, and its installation is an involved task. You should not attempt a manual installation of DirectX 3.

In addition, DirectSetup provides an automated way to install the appropriate Microsoft® Windows® registry information for applications that use the DirectPlayLobby object. This registry information is required for the DirectPlayLobby object to enumerate and start the application.

Although DirectSetup provides three API functions, only two are useful to you as an application developer. The other, **DirectXDeviceDriverSetup**, is designed for those who plan to install their own DirectX device drivers and is not addressed in this document.

DirectSetup Overview

This section contains general information about the DirectSetup component. The following topics are discussed:

Using the DirectXSetup Function

Preparing a DirectX Application for Installation

Enabling AutoPlay

Using the DirectXSetup Function

Applications and games that depend on DirectX use the **DirectXSetup** API function to install their system components on top of an existing Windows installation. This function updates the display and audio drivers to support DirectX (if required) during the DirectX installation process. **DirectXSetup** is provided to each application from the Dsetup.dll that is included with your product. You can find the declarations for DirectSetup in Dsetup.h.

Applications that use **DirectXSetup** must distribute the entire contents of the Redist directory, not just the contents of the \Redist\Directx directory. Although **DirectXSetup** allows you to install a single DirectX component, such as DirectDraw®, it is not recommended; the space-saving advantages are minimal because of the interdependent design of DirectX components. Applications must distribute the entire DirectX system, even if they use only one DirectX component.

Preparing a DirectX Application for Installation

When you are ready to create a way to install your application and the DirectX files on a user's system, you need to create a setup program that lists the files in your application, determines the amount of disk space required, and loads the appropriate DirectX files. You also need to create a directory on your distribution medium. You will place all the application's files and any additional DirectX components in this directory. The following topics describe these steps:

- *Creating the Setup Program*
- *Setting Up the Application Directory*

Creating the Setup Program

Included with this SDK is an example setup program that you can use as a model for your application. The setup program, called Dinstall and located in the \Dxsdk\Sdk\Samples\Setup directory, demonstrates how to load a sample program called Rockem in a selected directory. It also demonstrates one way to configure the **DirectXSetup** function.

The following steps describe how you can modify the Dinstall.c program to work for your application:

1. In an editor, open Dinstall.c.
2. You need to provide a list of your application's files that you want to load on a user's system. To modify the list in the Dinstall.c file to fit your application's needs, search for "copy_list", and then change the list of files in this structure to the list of files for your application.

   Dinstall installs files only in the default directory. If you want some of your application's files to be installed in a subdirectory, you need to modify Dinstall.c.
3. The Dinstall.c program does not determine whether enough free hard disk space is available on the user's system to successfully install your application. You should, however, add this functionality by writing your own code. To find the two locations in the Dinstall.c file to insert this code, search for "IDS_DISK_MSG".
4. The *lpszRootPath* parameter of **DirectXSetup** specifies the path to the Dsetup*.dll files (Dsetup.dll, Dsetup6e.dll, Dsetup6j.dll, Dsetupe.dll, and Dsetupj.dll) and the Directx directory on your distribution media. These dynamic-link libraries and this directory should be located in the same directory as the Dinstall executable (after it is compiled), unless there is an overwhelming reason to do otherwise. If all these files and directories are located in the same directory, the value of the *lpszRootPath* parameter should be set to NULL. This ensures that if the path changes when the files are placed on a compact disc or floppy disks from the root of the application, **DirectXSetup** will still function properly.

For example, suppose Dinstall.exe, Dsetup*.dll, and the Directx directory are located in an application directory called D:\Funstuff during the testing phase. Then, when you burn the files on a compact disc, suppose you put them in the root. If the *lpszRootPath* parameter is set to "\FUNSTUFF", the setup program (Dinstall.exe) will not function from the compact disc. However, if the *lpszRootPath* parameter is set to NULL, the setup program will function in both cases, because the path to Dsetup*.dll, and the Directx directory are still in the current directory.

If you decide to place the Dsetup*.dll files and the Directx directory somewhere other than in the directory that contains Dinstall.exe, you must pass the correct parameters to **DirectXSetup** and load Dsetup.dll correctly. The *lpszRootPath* parameter of **DirectXSetup** should contain the full path to Dsetup.dll. In addition, you need to use the **LoadLibrary** and **GetProcAddress** Win32® functions in your setup program to locate Dsetup.dll.

The content of the Setup dialog box is determined by data supplied in the Dinstall.rc resource file. To display your application's name and graphics, make the following changes to this resource file:

1. In an editor, open Dinstall.rc.
2. Search for all occurrences of "Rockem" and change them to the name of your application.
3. The graphics that are displayed in the Setup and Reboot dialog boxes are called Signon.bmp and Reboot.bmp in the resource file. You can either name your bitmap files these names, or you can change the names in the resource file to match the names of your bitmaps.
4. The icon for the Dinstall executable is called Setup.ico in the resource file, and it is specified by SETUP_ICON. You can either name your icon file Setup.ico, or you can change the name in the resource file to match the name of your icon file.
5. Optionally, you can change the default directory your application is installed in. To do this, search for "IDS_DEFAULT_GAME_DIR" (it is located in two places in the resource file) and change the path of the default directory.

After you have modified the Dinstall.c and Dinstall.rc files to fit your application's needs, you can compile them into the Dinstall.exe executable. You can also rename this executable (to Setup.exe, for example).

Setting Up the Application Directory

Before you commit your application to a compact disc or floppy disks, you should create an application directory to test your setup program. The application directory should contain all your application files, the setup program, and the DirectX files and drivers.

To set up the application directory, carry out the following steps:

Note

1. Create a directory that includes all your application's files. Be sure to create any subdirectories if needed, and place the appropriate application files in the subdirectories.
2. Copy the setup executable you wrote to the root of your application directory.
3. At the MS-DOS prompt, use the **xcopy** command to copy the Redist directory on the DirectX 3 compact disc to the root of your application directory. For example, if your application's root directory is D:\Fungame, and the E: drive is your CD-ROM drive, type the following:

   xcopy /s e:\redist\*.* d:\fungame

The root of your application directory should include the entire contents of the Redist directory distributed on the DirectX 3 SDK to ensure that the **DirectXSetup** function and the Dxsetup.exe file work properly.

After you copy all the appropriate files to the root application directory, it will look similar to this:

```
 Volume in drive D is SYSTEM
 Directory of D:\FUNGAME

.              <DIR>          07-26-96  6:43a .
..             <DIR>          07-26-96  6:43a ..
directx        <DIR>          07-26-96  6:43a directx
dsetup   dll          22,016  07-26-96  4:38a dsetup.dll
dsetup6e dll          36,224  07-26-96  4:38a dsetup6e.dll
dsetup6j dll          36,224  07-26-96  4:38a dsetup6j.dll
dsetupe  dll          42,496  07-26-96  4:38a dsetupe.dll
dsetupj  dll          42,496  07-26-96  4:38a dsetupj.dll
dinstall dll         168,960  07-26-96  4:38a dinstall.dll
yourfile exe          96,442  07-26-96  4:39a yourfile.exe
yourfile dat       1,508,228  07-26-96  4:39a yourfile.dat
...
```

Enabling AutoPlay

If you are building an AutoPlay compact disc title, you can copy the Autorun.inf file in the root directory of the DirectX 3 SDK compact disc to the root of your application directory. This text file contains the following information:

```
[autorun]
OPEN=SETUP.EXE
```

If your application's setup program is called Setup.exe, you will not have to make any changes to this file; otherwise, edit this file to contain the name of your setup program. For more information about the Autorun.inf file, see *The Autorun.inf File*.

DirectSetup Reference

Functions

DirectXRegisterApplication

```
int WINAPI DirectXRegisterApplication(HWND hWnd,
    LPDIRECTXREGISTERAPP lpDXRegApp);
```

Registers an ISV's game as an application designed to work with DirectPlayLobby.

- Returns TRUE if successful, or FALSE otherwise. If FALSE is returned, use the **GetLastError** Win32 function to get extended error information.

hWnd
Handle of the parent window. If this parameter is set to NULL, the desktop is the parent window.

lpDXRegApp
Address of the **DIRECTXREGISTERAPP** structure that contains the registry entries. These entries need to be filled in.

DirectXSetup

```
int WINAPI DirectXSetup(HWND hWnd, LPSTR lpszRootPath,
    DWORD dwFlags);
```

Installs one or more DirectX components.

- Returns SUCCESS if successful, or an error otherwise. For a list of possible return codes, see *DirectSetup Return Values*.

hWnd
Handle of the parent window for the setup dialog boxes.

lpszRootPath
Address of a string that contains the root path of the DirectX component files. This string must specify a full path to the directory that contains the Dsetup.dll file. (This directory is typically Redist.) If you are certain the current directory contains Dsetup.dll and the Directx directory, this parameter can be NULL.

dwFlags
One or more flags used to indicate which DirectX components should be installed. A full installation (DSETUP_DIRECTX) is recommended.

DSETUP_D3D	Installs Direct3D™.
DSETUP_DDRAW	Installs DirectDraw.
DSETUP_DDRAWDRV	Installs DirectDraw device drivers.
DSETUP_DINPUT	Installs DirectInput™.
DSETUP_DIRECTX	Installs all DirectX components.
DSETUP_DIRECTXSETUP	Installs **DirectXSetup** DLLs.
DSETUP_DPLAY	Installs DirectPlay®.
DSETUP_DPLAYSP	Installs DirectPlay service providers.
DSETUP_DSOUND	Installs DirectSound®.
DSETUP_DSOUNDDRV	Installs DirectSound device drivers.
DSETUP_DVIDEO	Installs DirectVideo.
DSETUP_PROMPTFORDRIVERS	Prompts before replacing display and audio device drivers.
DSETUP_RESTOREDRIVERS	Restores display and audio drivers.

Before you use **DirectXSetup** in your setup program, you should ensure that there is at least 5 MB of available disk space on the user's system. This is the maximum space required for DirectX to set up the appropriate files. If the user's system already contains the DirectX files, this space is not needed.

Structure

DIRECTXREGISTERAPP

```
typedef struct _DIRECTXREGISTERAPP {
    DWORD  dwSize;
    DWORD  dwFlags;
    LPSTR  lpszApplicationName;
    LPGUID lpGUID;
    LPSTR  lpszFilename;
    LPSTR  lpszCommandLine;
    LPSTR  lpszPath;
    LPSTR  lpszCurrentDirectory;
} DIRECTXREGISTERAPP, *PDIRECTXREGISTERAPP, *LPDIRECTXREGISTERAPP;
```

Contains the registry entries needed for applications designed to work with DirectPlayLobby.

dwSize
Size of the structure.

dwFlags
Reserved for future use.

lpszApplicationName
Name of the application.

lpGUID
Globally unique identifier (GUID) of the application.

lpszFilename
Name of the executable file to be called.

lpszCommandLine
Command-line arguments for the executable file.

lpszPath
Path of the executable file.

lpszCurrentDirectory
Indicates the current directory. This is typically the same as **lpszPath**.

Return Values

The **DirectXSetup** function can return the following values.

DSETUPERR_BADSOURCESIZE
A file's size could not be verified or was incorrect.

DSETUPERR_BADSOURCETIME
A file's date and time could not be verified or were incorrect.

DSETUPERR_BADWINDOWSVERSION
DirectX does not support the Windows version on the system.

DSETUPERR_CANTFINDDIR
The setup program could not find the working directory.

DSETUPERR_CANTFINDINF
A required .inf file could not be found.

DSETUPERR_INTERNAL
An internal error occurred.

DSETUPERR_NOCOPY
A file's version could not be verified or was incorrect.

DSETUPERR_NOTPREINSTALLEDONNT
The version of Windows NT on the system does not have all the DirectX 3 components installed.

DSETUPERR_OUTOFDISKSPACE
The setup program ran out of disk space during installation.

DSETUPERR_SOURCEFILENOTFOUND
One of the required source files could not be found.

DSETUPERR_UNKNOWNOS

The operating system on your system is not currently supported.

DSETUPERR_USERHITCANCEL

The Cancel button was pressed before the application was fully installed.

SUCCESS

A 0 is returned if the setup was successful and no restart is required.

A 1 is returned if the setup was successful and a restart is required.