Installing the DirectX SDK

Compilers

There are 2 major choices that I'm aware of:
- Microsoft Visual Studio (C++/C#/whatever)
- Dev C++

Borland is another choice; there is a website that will help you get the SDK setup for this compiler (http://www.geocities.com/foetsch/bcbfaq/bcbddraw.htm).

In our lectures we will be using Microsoft Visual Studio .NET 2003, it uses the same C++ libraries as VSC++ 7.0

I've never used Dev C++, but you can get it here (http://www.bloodshed.net/).
It's free, which is nice. I haven't used it myself so I don't have an opinion on it.

Microsoft Visual C++ is the path of least resistance. DirectX 9 only supports version 6 or greater.

DirectX SDK

The Software Development Kit for DirectX is really big. You do need it though, so if you're not on a high-speed connection you may want to use a download manager and leave it running over night.

You can download this direct from Microsoft:
http://msdn.microsoft.com/library/default.asp?url=/downloads/list/directx.asp

Compiler Install

This is the first step you should do. Depending on which product and which version you're installing, it can be quite different. But it should be pretty straight forward regardless.

SDK Install

Again, this is pretty simple too. I want to emphasize that when you install it you will be asked if you want to install the Retail or Debug version. **Choose the Debug version.** The value of using the Debug libraries cannot be emphasized enough. Also, make a note of where you install the SDK; you'll need the path information soon.

DirectX 9.0 SDK is a bit temperamental and may waste some of your time, if you let it. The following tips should help you avoid wasting time.

If you haven't already done so, update your Windows system with the latest patches from Microsoft. Next, download the latest release of the .NET Framework SDK, install it, and check for updates with Windows Update and manually, in MSDN's .NET Framework section. Now you can download your choice of the DirectX 9.0 SDK versions (full, C#, or VB.NET) and double-click on it to unpack the installation files to a temporary directory.

When you run setup, it will ask you for a path to install the SDK. There is a bug in the installer that, in some cases, does not accept the path you choose when you click on the Browse button. If the problem persists and you cannot force the installer to obey you, the solution to this problem is to quit the setup, remove temporary files, unpack the archive again, and then, when the installer asks for the path, change C:\DXSDK to a different path (e.g., E:\DXSDK) manually, by typing it into the Path text field.

Another strange thing that you might run into is a problem with DLL locations. The DirectX DLLs are put into the C:\WINDOWS\assembly\ directory, where they are essentially inaccessible to your development environment, be it Visual C# or Visual Basic .NET. If you open up a windows browser and go to the C:\WINDOWS\assembly\ directory you should see the followin components:
Microsoft.DirectX
Microsoft.DirectX.AudioVideoPlayback
Microsoft.DirectX.Diagnostics
Microsoft.DirectX.Direct3D
Microsoft.DirectX.Direct3DX
Microsoft.DirectX.DirectDraw
Microsoft.DirectX.DirectInput
Microsoft.DirectX.DirectPlay
Microsoft.DirectX.DirectSound

It doesn't always happen, but if you open a sample project from the DirectX 9.0 SDK and you see exclamation marks in the References list next to Microsoft.DirectX libraries, then that's the sign of this particular installer misbehaviour. Fortunately, it is very easy to fix this problem. Simply open the command line (console) window and type:

```
cd c:\windows\assembly\
dir
```

and you should see this:

```
Microsoft.DirectX
Microsoft.DirectX.AudioVideoPlayback
Microsoft.DirectX.Diagnostics
Microsoft.DirectX.Direct3D
Microsoft.DirectX.Direct3DX
Microsoft.DirectX.DirectDraw
Microsoft.DirectX.DirectInput
Microsoft.DirectX.DirectPlay
Microsoft.DirectX.DirectSound
```

The DLLs we are looking for are located in subdirectories with long names like:

```
Microsoft.DirectX.Direct3D\1.0.900.31bf3856ad364e35\
   Microsoft.DirectX.Direct3D.dll
```

Create a normal folder in your chosen location and copy each DLL to it. Then, update the References in your project and Build a sample application. Everything should work fine now.

Note: It may store the .DLL components in the C:\WINDOWS\assembly\GAC folder instead of the base assembly folder.

Note: If you downloaded the DirectX 9.0 Redistributable, do not install it, but keep it on CD-R until you have your application ready for distribution. It contains files that Microsoft lets developers distribute to end users, and is of not much use to programmers. It's the same story with the NET Framework Redistributable.

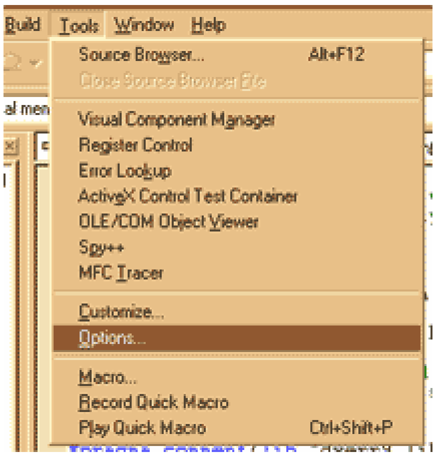
What You Need To Do

Configuring the compiler is really easy. The hardest part is figuring what you need to do and where to do it. The following sections will show step-by-step what you have to tweak and where it is.

VSC++ 6.0:

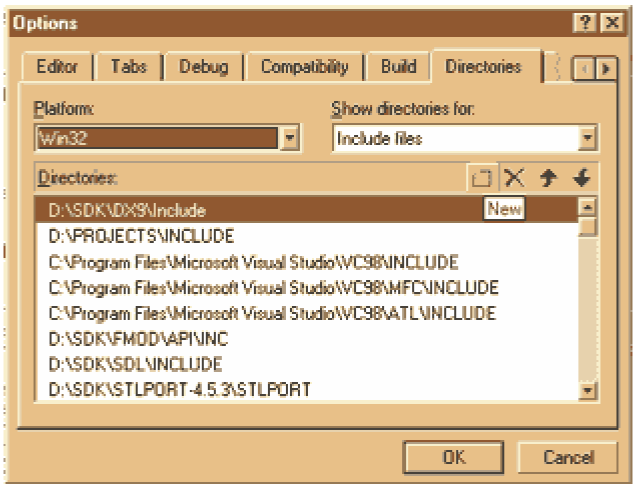
Tools

Fire up VC6 and select Options from the Tools menu:



Directories

Select the Directories tab. Make sure Show directories for is set to "Include files".



Platform: Win32

Show directories for: Include files

Directories:

```
D:\SDK\DX9\Include
D:\PROJECTS\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\MFC\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\ATL\INCLUDE
D:\SDK\FMOD\API\INC
D:\SDK\SDL\INCLUDE
D:\SDK\STLPORT-4.5.3\STLPORT
```

Select the New button and then navigate to the Include directory in your SDK install. I installed the SDK at D:\SDK\DX9 so my Include directory is D:\SDK\DX9\Include.

Once you've done that, use the Up arrow button to move that entry to the top of the list. When VC++ looks for a header it starts at the top and moves down through the list until the header is found. Putting the SDK directory at the top means it will find the latest headers rather than the ancient ones included with VC++ 6.

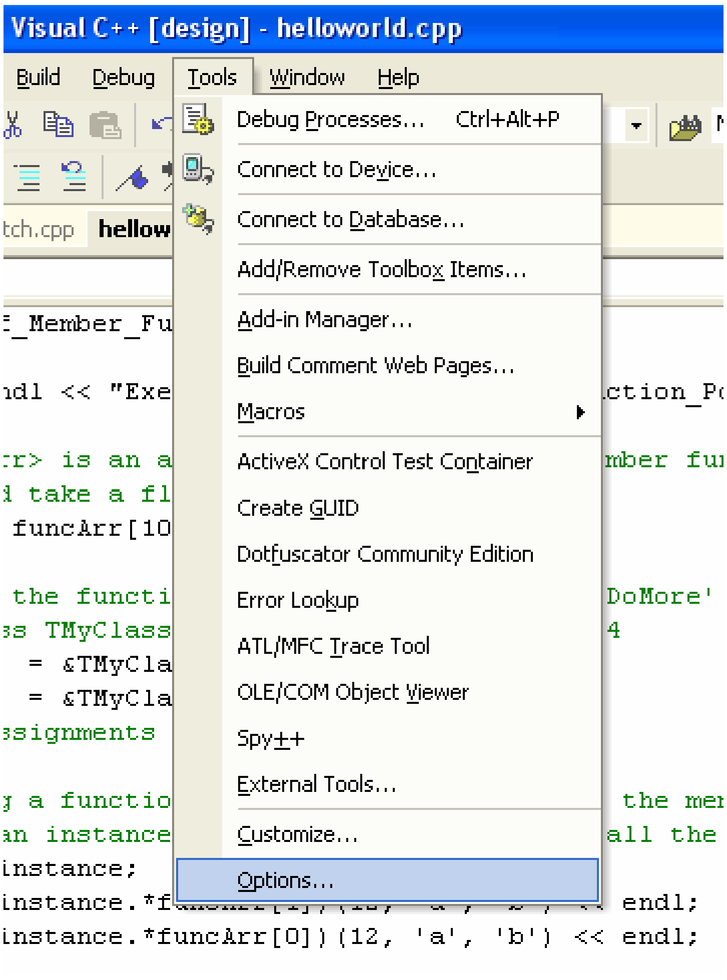
Now change Show directories for to "Library files". Add the path to the "Lib" directory in your SDK install and move it up to the top.

That's it! Your compiler is now configured and ready to go.

The process for Microsoft Visual Studio .NET 2003 is similar; I'll show you how here:

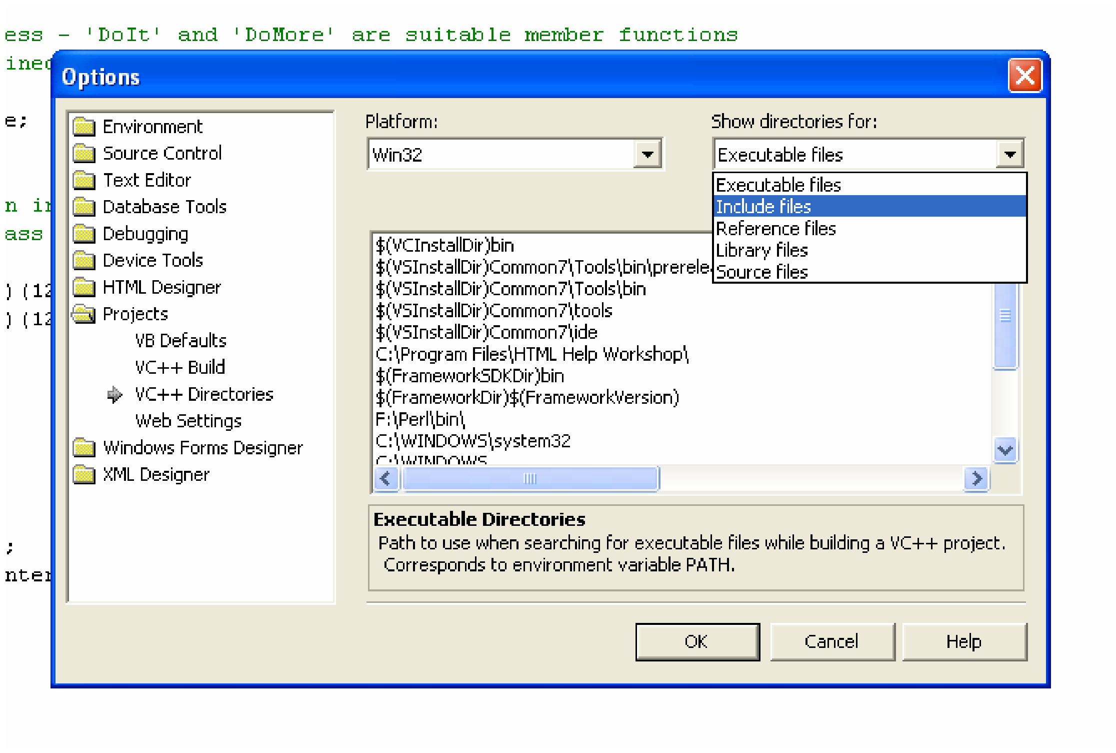
VS .NET 2003:

Tools
Again fire up the IDE



Directories
Select the Directories tab. Make sure Show directories for is set to "Include files".

Now it gets a little more complicated than last time as you have to select the relevant file from the list on the left hand side, other than that it is almost identical.



```
ess - 'DoIt' and 'DoMore' are suitable member functions
ined
e;
n i
ass
) (12
) (12
;
nte
```

```
$(VCInstallDir)bin
$(VSInstallDir)Common7\Tools\bin\prerele
$(VSInstallDir)Common7\Tools\bin
$(VSInstallDir)Common7\tools
$(VSInstallDir)Common7\ide
C:\Program Files\HTML Help Workshop\
$(FrameworkSDKDir)bin
$(FrameworkDir)$(FrameworkVersion)
F:\Perl\bin\
C:\WINDOWS\system32
C:\WINDOWS
```

Executable Directories
Path to use when searching for executable files while building a VC++ project. Corresponds to environment variable PATH.

<u>Setting up Debug</u>

The debug version of DirectX (you did install the Debug version, right?) is an invaluable tool. Often finding the cause of a bug is as easy as viewing the Debug output generated by the Debug libraries. Both errors and warnings are output and paying close attention to them will help you make a solid, well performing application.

The output of the debug libraries is sent to the same place as anything passed to the OutputDebugString Win32 function. If you use Microsoft Visual C++, there is a debug window as part of the IDE. When you run an app from within the IDE, the debug window is shown at the bottom of the IDE window. If you do not have MSVC++ or want/need to run your app outside of the IDE, then you can download DebugView from SysInternals. It's completely free and very handy to have around. Debug View: http://www.sysinternals.com/ntw2k/freeware/debugview.shtml

The Control Panel

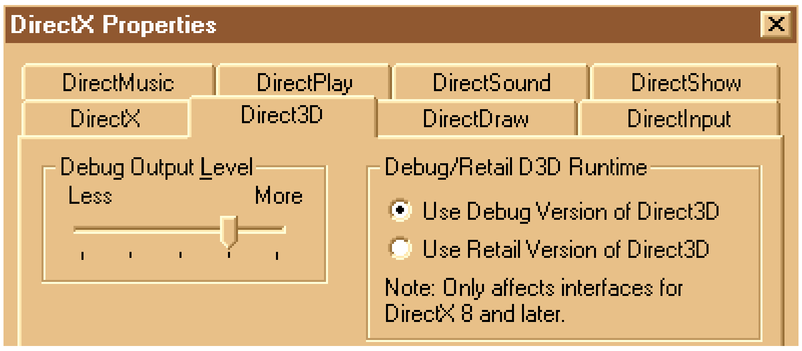
If you've installed the Debug version of DirectX then it will have added an icon to your control panel. Run the application and you will see a window with 1 tab for each major component of DirectX. On these tabs are various controls that will assist you in debugging your DirectX applications.

Select the Direct3D tab. There are a number of things presented here, but we're mostly interested in 2 of them: Debug Output Level and Debug/Retail D3D Runtime.

Debug Output Level, as you would probably guess, regulates how verbose the messages it sends are. Set it to the lowest level and very few messages will be sent.

Set it to the highest level and you will be swimming in output (hence the term "Debug Spew", which is often used to refer to this debug output). Setting it somewhere in the middle is a good default.

Debug/Retail D3D Runtime chooses whether applications will use the Retail or Debug libraries when they run. This allows you to switch to the more optimized Retail libraries for profiling or gaming, while being able to quickly switch back to Debug for development.


DirectX Properties

DirectMusic | DirectPlay | DirectSound | DirectShow | DirectX | Direct3D | DirectDraw | DirectInput

Debug Output Level
Less — More

Debug/Retail D3D Runtime
(●) Use Debug Version of Direct3D
( ) Use Retail Version of Direct3D
Note: Only affects interfaces for DirectX 8 and later.

Managed DirectX
If you are using Managed DirectX there is one thing you need to do in addition to the steps listed above. Go to the project settings and under Debugging, set Enable Unmanaged Debugging to true. Without this the Debug output will not be visible.

D3DX
All of this works well for the standard DirectX DLLs since they're dynamically linked, but what about D3DX? In the Lib directory of the SDK you will see 3 different versions of the D3DX9 library. There is a retail library, a debug library, and a debug library that actually uses a DLL. I've never really understood the point of that last one, so we'll focus on the first two.

When you link with the debug version of D3DX, you get the same benefits of debug output as with the standard DirectX libraries. Typically you'll want to link with the debug library in your debug builds, and the retail library in your retail builds. Since I use the #pragma comment mechanism to link libraries, all it takes is a little #ifdef magic and you're done. Here's the code snippet:

```
#ifdef _DEBUG
  #pragma comment(lib,"d3dx9d.lib")
#else
  #pragma comment(lib,"d3dx9.lib")
#endif
```