Hooks Unsnagged

by Warren Kovach

In Issue 26 I described how to use keyboard hooks to trap system-wide keystrokes. The code in that article formed the core of my shareware program Accent Composer. Over the past few months I've been getting reports of strange errors that seemed to be linked to Accent Composer, but were occurring in other programs and at odd times. Further investigation showed that the errors occurred in *all* hook programs written in Delphi. The problems were finally solved (with help from Colin Messitt of Oakley Data Services).

The first problem is that any Windows hook compiled under Delphi 3 and C++Builder will conflict with Windows 95 password dialog boxes. These include network logon, Internet dialup and screen saver password dialogs. Sometimes when a password is being typed there will be one or more occurrences of runtime error 216 (access violation). More often a page fault or access violation exception will occur in Windows Explorer, MPREXE.EXE (the network multiple protocol router) or some other module. These exceptions usually occur about a minute or so after the password has been typed, although sometimes they are delayed until the hook program is exited or Windows is shut down.

This problem seemed resistant to virtually every attempt to fix it. The only clue was that hooks compiled with Microsoft's C compilers did not exhibit this problem. Eventually I discovered that if both the DLL and the hook host program were recompiled under Delphi 2 the problem went away. I suspect there is something about the code produced by the new back-end compiler shared by Delphi 3 and C++Builder that causes conflicts with the password routines. I've reported this to Borland so hopefully they will be able to track down the cause and fix it in the next version. For now, you should write your Windows hooks with Delphi 2!

The second problem also occurs in all Delphi hooks, even Delphi 2. It reportedly can occur in hooks produced with other non-Microsoft compilers as well, but not those compiled with Microsoft tools. This is a conflict with 3D graphics programs using the OpenGL libraries. If a Windows hook is active, and an OpenGL-based program is in use, it will eventually crash, usually during rotation or translation operations, with a floating point error. This is actually a problem in OpenGL: Microsoft is aware of this and says it will be fixed in future versions or service packs.

The problem is that the floating point operations in OpenGL can sometimes perform invalid operations, such as division by zero. Microsoft's compilers mask floating point exceptions in these circumstances so that they are effectively ignored. Other vendors' compilers are not so forgiving, however, and allow these errors to raise exceptions. You can easily mask the exceptions yourself using the `Maskx86Exceptions` function in Listing 1.

This function should be called each time a new process attaches to a hook DLL. This can be done using the `System` unit's `DLLProc` variable, which is pointed at the `DLLMain` procedure in the library's initialization code. This then calls `LibEnter` each time a process attaches to the DLL and `LibExit` when it detaches. The exception mask is set back to its previous state on detachment.

These floating point errors can also occur in Delphi programs that use the OpenGL libraries directly, not just in hooks. If you are doing OpenGL programming you may need to bracket your drawing code with the `Maskx86Exceptions` and `RestoreX86Mask` routines.

Warren Kovach is the author of *Delphi 3 – User Interface Design*, published by Prentice Hall. You can contact Warren at wlk@kovcomp.co.uk or by visiting http://www.kovcomp.co.uk/

➤ *Listing 1*

```
library MyHook;
{ ... full code on disk }
var dwOldMask: Pointer;
begin
  asm
    fnstcw WORD PTR dwOldMask;
    mov eax, dwOldMask;
    or eax, $3f;
    mov WORD PTR dwOldMask + 2, ax;
    fldcw WORD PTR dwOldMask + 2;
  end;
  result := dwOldMask;
end;
procedure RestoreX86Mask(dwOldMask: Pointer);
begin
  asm
    fnclex;
    fldcw WORD PTR dwOldMask;
  end;
end;
{ your hook procedure here }
```

```
procedure LibEnter;
begin
  P86Mask := MaskX86Exceptions;
end;
procedure LibExit;
begin
  Restorex86Mask(P86Mask);
end;
procedure DLLMain(Reason:DWORD);
begin
  case Reason of
    DLL_PROCESS_DETACH : LibExit;
    DLL_PROCESS_ATTACH : LibEnter;
    DLL_THREAD_ATTACH  : ;
    DLL_THREAD_DETACH  : ;
  end;
end;
begin
  DLLProc := @DLLMain;
  DLLMain(DLL_PROCESS_ATTACH);
end.
```