**19.2 Windows and UNIX Portability**

When writing samples and prototype code and even production applications, keep in mind that different UNIX implementations and Windows 95/NT have different APIs, provide different system services, and can even provide substantially different development environments (such as contents of include files, location of libraries, etc.). Here are a few things to look out for when writing a program under UNIX with the intent to port to Windows or other UNIX operating systems:

* The Win32 versions of the OpenGL and GLU header files depend on macros defined by including the <windows.h> header file. This forces you to do the following for Win32 portability:[10](http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/footnode.html" \l "foot20957)
* #ifdef \_WIN32
* #define WIN32\_LEAN\_AND\_MEAN /\* somewhat limit Win32 pollution \*/
* #include <windows.h>
* #endif
* #include <GL/gl.h>
* #include <GL/glu.h>

Unfortunately, including <windows.h> has the unfortunate side effect of introducing literally thousands of macros and type declarations into your compilation environment. This undesirable ``name space pollution'' can sometimes affect source code portability by conflicting with your program's own macros and types. This can particularly be a problem for UNIX programmers that are not familiar with all the junk that comes with including <windows.h>.

One alternative is to include the <GL/glut.h> header. The GLUT header automatically includes <GL/gl.h> and <GL/glu.h> and guarantees to include these headers in a way that avoids introducing the name space pollution of including <windows.h>. If you use GLUT, your programs will automatically be more portable by simply including <GL/glut.h> and *not including* <GL/gl.h> or <GL/glu.h> directly (simply letting <GL/glut.h> include them).

* Avoid the identifiers near and far, which are reserved words in most Intel and Windows compilers. Common replacements are nnear and ffar.
* The math constant M\_PI and related constants such as M\_PI\_2 are not provided by at least one Win32 development environment. You may find adding the following code after #include <math.h> to be helpful:
* #ifndef M\_PI
* #define M\_PI 3.14159265358979323846
* #endif
* Do not #include <unistd.h> as it contains UNIX-specific definitions. At the very least, check with your Windows environment before using functions or constants from <unistd.h>.
* The ANSI C library defined constants EXIT\_SUCCESS and EXIT\_FAILURE may not be available. You could include code to define these constants similar to the above code for M\_PI.
* Single-precision versions of trigonometric functions such as sinf and cosf while desirable for performance may not be available on all platforms.
* If you need to generate random numbers, use the ANSI C function rand. Do not use the traditional UNIX functions random or drand48 since they are not supported by Win32.
* When opening binary files such as an image file with fopen, be sure to use a mode string of "wb" for writing or "rb" for reading. Without the b option (it stands for binary), Win32 opens the file for access as a text file and performs translations of formfeed and linefeed characters. Using the b option suppresses these translations.
* If you use the GLU tessellator, use the CALLBACK calling convention identifier (this is really just a macro for the calling convention keyword \_\_cdecl in Win32). You must do this for the callbacks to work correctly under Win32. For the benefit of UNIX environments that do not define a CALLBACK macro (because they do not need it!), after including <GL/glu.h>, include the following:
* #ifndef CALLBACK
* #define CALLBACK
* #endif

Then using a GLU tessellator begin callback as an example say:

static void CALLBACK

begin(GLenum type, void \*polyData)

{

glBegin(type);

}

When registering the callback, say:

gluTessCallback(tess, GLU\_TESS\_BEGIN\_DATA,

(void (CALLBACK\*)()) &begin);

This advice also applies to the other GLU routines that require callback functions to be supplied. These other routines are gluQuadricCallback() and gluNurbsCallback().

* It is sometimes tempting in graphics programs to name a variable quad, short for quadrilateral. Avoid the temptation. Some operating systems such as IBM's AIX variant of UNIX define quad to be 64-bit data type in <sys/types.h> which is often implicitly included by many system header files.

A more in-depth list of portability considerations is available in the file Portability.txt in the GLUT source code distribution. GLUT is described in more detail in Appendix [B](http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node398.html#GLUT).