Release Notes for Wild Magic Version 3.0

David Eberly
Magic Software, Inc.
http://www.magic-software.com

Created: October 7, 2004

Contents

1 About this Document

Please read this document before attempting to compile and run the libraries and applications! Each supported platform (Microsoft Windows, Linux, Macintosh) has prerequisites that must be met in order for Wild Magic Version 3 (WM3) and its tools to compile, link, and run successfully. Once the prerequisites are met, the projects must be compiled in a particular order. Standard support questions are about issues that arise when the prerequisites are not met or when the projects are compiled in the wrong order.

Other information of interest is available here, so browse the entire document before starting the installation. Generally, you should also check the official Wild Magic web site, `http://www.wild-magic.com`, for updates, bug fixes, new features, and other materials. The update history page always has a date for the last modification, so you should be able to track what you have or have not downloaded.

2 Prerequisites and Portability

WM3 is architected to be portable. Generally, no library will automatically compile, link, and run when placed on a new platform. If you attempt to compile WM3 on platforms other than the ones we tested, there may be minor issues that need to be resolved; for example, there may be makefile issues or no GLUT support on the platform. Wild Magic does not currently have a layer that directly supports X Windows. Some of the tools only work on the Microsoft Windows platform; for example, Max and Maya are modeling packages that run on Windows–the exporters naturally work only with those packages.

The platforms and environments on which we have tested Wild Magic 3 are listed in the next subsections. Any prerequisites are also mentioned.

2.1 Microsoft Windows XP

2.1.1 Compilers

Developer Studio. We use Microsoft Visual Studio .NET 2003 (MCVC7.1). The CD-ROM contains project files for this platform (`*.vcproj` files). Many developers are still using Microsoft Visual C++ version 6 (MSVC6). Even though the compiles are slower with MCVC7.1, we recommend upgrading because we find the development environment to be a lot better. Some MSVC6 problems have been fixed in MCVC7.x (for example, the nasty STL xtree static member bug). Moreover, Microsoft has officially stopped supporting MSVC6. Although the CD-ROM does not contain MSVC6 project files (`*.dsp` files), the WM3 web site does have these. They were not ready in time for the delivery deadline of the CD-ROM master for the *3D Game Engine Architecture* book.

If you have only Microsoft Visual Studio .NET 2002 (MSVC7.0) and try to open the `vcproj` files, you will get an error about the incorrect version. A hack to work around this is to edit the `vcproj` file (it is ASCII) and replace the line

```
Version="7.10"
```

by

```
Version="7.00"
```

This works for all the Wild Magic project files, but we cannot guarantee this will work for other projects.

An installation of MSVC7.x will associate itself with the `.dsp` files and the `.dsw` files of MSVC6. If you prefer to use both compilers on your machine, restore the association of Version 6 with `.dsp` and `.dsw` to avoid Version 7.x trying to automatically convert the project files.

MinGW. At one time we tried to compile using MinGW (Minimalist GNU for Windows), available from `http://www.mingw.org`, but the package had trouble with our makefiles. In particular, the make system was claiming certain subdirectories in our distribution did not exist when in fact they did. We had not downloaded the full distribution and suspect that the MSYS package does not have this problem. If enough requests occur to support MinGW, we will consider adding support for this package.

C++ Builder. Some users of Wild Magic Version 2 reported compile problems with Borland's C++ Builder Studio, `http://www.borland.com/cbuilder/`. We were able to suggest work arounds for most of these. Some of the reported errors were clearly bugs in the compiler, but we were unable to find work arounds.

2.1.2 DirectX

The DirectX renderer uses the DX9.0c SDK. If you plan on compiling the renderer project

```
MagicSoftware/WildMagic3/Renderers/DirectX/DirectX.vcproj
```

you must have the DirectX SDK installed on your machine. You may download this from Microsoft's web site, `http://www.microsoft.com`. You should see a link for "Downloads" on the home page. The location for the DirectX downloads might change over time, so you will have to browse the Microsoft site to find them. If you have the DX9.0a SDK, you will have to enable a preprocessor define in the file

```
MagicSoftware/WildMagic3/Renderers/DirectX/Wm3DrawText.cpp
```

The comments at the top of this file indicate that the function prototypes changed between DX9.0a and DX9.0b for D3DXCreateFont and the DrawText member of the ID3DXFont interface.

If you install the DirectX SDK, you should allow the DX9.0c installer to modify the global search paths in the Developer Studio IDE. We used the default locations,

```
C:\Program Files\Microsoft DirectX 9.0 SDK (Summer 2004)\{Include,Lib}
```

If you did not allow the installer to modify your global paths, you must modify the include and library paths in the DirectX project settings. The application project,

```
MagicSoftware/WildMagic3/Application/Application.vcproj
```

has configurations DebugDX and ReleaseDX to support applications using a DirectX renderer. The include and library paths must be modified to contain the locations where DirectX was installed.

2.1.3 OpenGL

The OpenGL renderer project is

```
MagicSoftware/WildMagic3/Renderers/OpenGL/OpenGL.vcproj
```

It uses GLEW for extension handling. This package is downloadable from `http://glew.sourceforge.net`. We downloaded version 1.1.4, but had to make some modifications to `glew.h` and `glew.c`. The source files from it that we use are

```
MagicSoftware/WildMagic3/Renderers/GLEW/glew.c (modified)
MagicSoftware/WildMagic3/Renderers/GLEW/glew.h (modified)
MagicSoftware/WildMagic3/Renderers/GLEW/glxew.h (original)
MagicSoftware/WildMagic3/Renderers/GLEW/wglew.h (original)
```

The file `glew.c` is duplicated in

```
MagicSoftware/WildMagic3/Renderers/OpenGL/glew.c (modified)
```

because the MSVC7.x projects expect to find all the project source in a single directory. The modifications in `glew.{h,c}` are wrapped by comments that contain `Begin Magic Software` and `End Magic Software`. The comments indicate what the reasons are for the changes. We use `glBlendColor` for texturing. The function pointer `glewBlendColor` was initialized in the section `_glewInit_GL_ARB_imaging`, but the blending is part of OpenGL 1.4 and needs to be initialized in the `_glewInit_GL_VERSION_1_4` section. Another change has to do with Apple's choice to access OpenGL header files through their framework mechanism rather than through the standard include path on Unix machines. For example, `GL/gl.h` does not work on the Macintosh, but `OpenGL/gl.h` does. The last change, no doubt, occurs in later versions of GLEW. In our version, we had to replace a line of code in `_glewInit` that compares a version number, `s[2]` to each of 1 through 4. The equality comparison to 4 was replaced by an inequality to allow OpenGL 1.5 or larger versions not to fail the compilation.

In order to use GLEW, you have to copy the header files

```
MagicSoftware/WildMagic3/Renderers/GLEW/glew.h
MagicSoftware/WildMagic3/Renderers/GLEW/glxew.h
MagicSoftware/WildMagic3/Renderers/GLEW/wglew.h
```

to directories that were set up when you installed Developer Studio. We install the compilers to the default directories. The targets for the copy are

```
// MSVC 6
C:\Program Files\Microsoft Visual Studio\VC98\Include\GL

// MSVC 7.0 (.NET 2002)
C:\Program Files\Microsoft Visual Studio .NET\Vc7\PlatformSDK\Include\gl

// MSVC 7.1 (.NET 2003)
C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include\gl
```

If you installed the compilers to different locations, you must find the appropriate directories to copy to.

The OpenGL project has Debug and Release configurations that use Windows OpenGL support (WGL). The project also has configurations DebugGLUT and ReleaseGLUT that use Mark Kilgard's GLUT package with a Win32 implementation by Nate Robins, `http://www.xmission.com/~nate/glut.html`. The version is 3.7.6. As per license agreement, the GLUT distribution is freely distributable. We have included that distribution,

```
MagicSoftware/WildMagic3/Renderers/GLUT/glut32.dll
MagicSoftware/WildMagic3/Renderers/GLUT/glut32.lib
MagicSoftware/WildMagic3/Renderers/GLUT/glut-3-spec.pdf
MagicSoftware/WildMagic3/Renderers/GLUT/glut.def
MagicSoftware/WildMagic3/Renderers/GLUT/glut.h
MagicSoftware/WildMagic3/Renderers/GLUT/README-win32.txt
```

GLUT is a wrapper around platform–dependent OpenGL window creation and destruction, but also has other basic capabilities to support some windowing system calls. The intent of GLUT is to provide a portable wrapper so that your application code can compile on any platform that has installed the GLUT headers and libraries.

If you plan on building the DebugGLUT and ReleaseGLUT configurations, you need to copy some of the GLUT files to the appropriate directories. Once again, we installed the compilers to the default directories. If you did not, you need to identify the target locations on your system. The paths below show the paths to the copied files.

```
// MSVC 6
C:\Program Files\Microsoft Visual Studio\VC98\Include\GL\glut.h
C:\Program Files\Microsoft Visual Studio\VC98\Lib\glut32.lib

// MSVC 7.0
C:\Program Files\Microsoft Visual Studio .NET\Vc7\PlatformSDK\Include\gl\glut.h
C:\Program Files\Microsoft Visual Studio .NET\Vc7\PlatformSDK\lib\glut32.lib

// MSVC 7.1
C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include\gl\glut.h
C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\lib\glut32.lib
```

Regardless of compiler version, copy the dynamic link library `glut32.dll` to the appropriate Window's system directory. For example, this might be named

```
C:\Windows\System32
```

Alternatively you can copy the DLL to any directory that your applications normally search for dynamic link libraries.

2.1.4 Cg Toolkit

If you are going to write shader programs to be used by WM3, you need to have nVidia's Cg Toolkit installed. You can download this from nVidia's web site, `http://www.nvidia.com`. At the home page, select the "Developers" link, then the "Tools & SDKs" link. Under "Software Developement" you should see the link for "Cg Toolkit". We had downloaded Version 1.1 of the toolkit. The CD-ROM contains a simple compiler to generate Wild Magic source code from Cg programs,

```
MagicSoftware\WildMagic3\Tools\CgToWm3\CgToWm3.vcproj
```

The project settings use the default paths for where the Cg Toolkit was installed,

```
C:\Program Files\NVIDIA Corporation\Cg\{include,lib}
```

If you installed the toolkit in a different location, you must modify the project settings accordingly.

2.2 Linux

We currently use Red Hat Linux version 8. The command `g++ -v` shows

```
gcc version 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
```

Makefiles are used to build the WM3 distribution. You should have a 3D graphics card installed in your machine with the appropriate drivers from your graphics card manufacturer. Such drivers can be downloaded from the manufacturer's web site. These are usually not "plug and play" and require some effort to install, but our experience with this has been not too painful.

If you do not have the GLUT distribution installed on your machine, you must do this since the WM3 distribution for Linux uses GLUT. Consult your Linux manuals to find out how to install GLUT.

We use GLEW for OpenGL extension handling. See Section 2.1.3 for details. You need to copy the header files

```
MagicSoftware/WildMagic3/Renderers/GLEW/glew.h
MagicSoftware/WildMagic3/Renderers/GLEW/glxew.h
MagicSoftware/WildMagic3/Renderers/GLEW/wglew.h
```

to the directory

```
/usr/include/GL
```

You probably need to be have root privileges to do this. The GLUT header file `glut.h` should already be in this directory.

2.3 Macintosh

You need to be running Macintosh OS X, preferably the latest version 10.3.5, and you should have a 3D graphics card installed. You should also have the Macintosh Developer Tools installed, preferably with Xcode Version 1.5. The CD-ROM comes with Xcode projects. You may use CodeWarrior Version 9, but you still need the various Frameworks provided with the Macintosh Developer Tools. The CodeWarrior project files are at the Wild Magic web site. They were not ready in time for the delivery deadline of the CD-ROM master for the *3D Game Engine Architecture* book.

We also use GLEW for OpenGL extension handling. See Section 2.1.3 for details. In fact, the extension handling is necessary because even though it appears the OpenGL provided by Apple is 1.4, there are certain extensions that made it into the 1.4 specifications, but the preprocessor defines are not found in the OpenGL headers provided by Apple. For example, we use vertex buffer objects. The defines for this were not present in the OpenGL headers.

You need to copy the header files

```
MagicSoftware/WildMagic3/Renderers/GLEW/glew.h
MagicSoftware/WildMagic3/Renderers/GLEW/glxew.h
MagicSoftware/WildMagic3/Renderers/GLEW/wglew.h
```

to the header subfolder of the OpenGL framework. The path is relative to the hard disk on which you installed the operating system. We chose to use a Terminal window to do the copy. Our path is

```
Mac:/System/Library/Frameworks/OpenGL.framework/Headers
```

This is actually a symbolic link, but that is irrelevant. If you perform an 'ls' command, you will see various headers including the OpenGL headers.

We install the WM3 distribution to the desktop. From a Terminal window for user 'joe', the path to the distribution is

```
/Users/joe/Desktop/MagicSoftware
```

From the Terminal window where we changed directory to the OpenGL headers, our first attempt to copy using the 'cp' command was

```
cp /Users/joe/Desktop/MagicSoftware/WildMagic3/Renderers/GLEW/*.h .
```

This failed because of not having root privileges, even though the user has administrator privileges. If this fails, you can try to "su root" first, the "cp". This also failed for us because of not having root privileges. What did work was

```
sudo cp /Users/joe/Desktop/MagicSoftware/WildMagic3/Renderers/GLEW/*.h .
```

When prompted for a password, enter your administrator password.

Alternatively, an administrator can give himself root privileges.

1. From the Finder Window toolbar, click "Go" to bring up a menu. Select "Applications" in that menu.
2. Locate the NetInfo Manager utility and open it.
3. If you see the path `/`, you have the correct window to work with. If you do not automatically see this, select "Domain" on the toolbar and select "Open". Select the default domain, which will be `/`.
4. Click on the lock button on the bottom-left side of the window to allow you to make changes. You will be prompted for the name and password of an administrative user. Enter these, then click OK.
5. From the NetInfo menu on the toolbar, select "Security" to bring up a submenu. If you see "Disable Root User", you already have root privileges. Otherwise, you will see "Enable Root User". A message will indicate that the password is blank. Click OK.
6. Enter the root password (twice, of course) you want to use, then click OK.

The Cg Toolkit from nVidia is available now for the Macintosh. You can download this from nVidia's web site, `http://www.nvidia.com`. At the home page, select the "Developers" link, then the "Tools & SDKs" link. Under "Software Developement" you should see the link for "Cg Toolkit". We have not yet ported the Wild Magic tool, `CgToWm3`, to the Macintosh.

2.4 Platform Differences

Platform differences are encapsulated in a small number of files. Inclusion of files or portions of files is controlled by preprocessor defines. In the core engine itself, the files

```
MagicSoftware\WildMagic3\Source\System\Wm3Platforms.h
MagicSoftware\WildMagic3\Source\System\Wm3System.cpp
```

encapsulate some basic system operations that are platform specific. The interface files

```
MagicSoftware\WildMagic3\Source\System\Wm3System.{h,inl}
```

are platform independent.

The Microsoft Windows platform requires that the preprocessor symbol `WIN32` be defined. The MSVC compilers already define these, so you do not need to add a define to the project settings. The Macintosh platform requires that `__APPLE__` be defined. Both Xcode and CodeWarrior automatically define this, so you do not need to add a define to the project settings. If neither `WIN32` nor `__APPLE__` are defined, the default is to compile for Linux. Naturally, as more platforms are added, the preprocessor mechanism in `Wm3Platforms.h` must be modified.

2.5 Explicit Instantiation of Templates

To support both `float` and `double` mathematical objects and functions, we use template classes that are *explicitly instantiated*. Some of these classes have static data members. Normally you need to be wary of using static data in templates. The problem is that if the template is expanded in a library, a static data

member is created in that library. An application that uses the template will expand the template code and cause another static data member to be generated. The intended semantics of the template are that exactly one static data member exists, so the generation of two is a real problem. Your complete application can wind up accessing one or the other static data member, potentially causing problems. This is the infamous bug in the MXVC6 template classes. The `xtree` class had a static data member that represented a "null object" (a null pointer, so to speak). The null object was internally modifiable. We discovered this problem when using the Wild Magic Version 1 dynamic link library, `MagicWM.dll`, in an application that was iterating over the elements of a standard container class, then crashed because the null object created in the application code was being used by the iterator, but the iterator should have used the null object created in the DLL.

If the static data member is constant, the existence of two copies is most likely not a problem. We have various static constant data members in the template classes. However, the template code itself is in `.inl` files that are accessed directly by source files in the Wild Magic library. These files are not exposed to an application through the template header files, so the application cannot implicitly instantiate the templates. Only the library itself can explicitly instantiate the templates and any static data members that go with them.

In our opinion, explicit instantiation of templates in the manner we have used is important for code reuse. We prefer not to have two copies of source code, one for `float` and one for `double`. Moreover, the template classes are designed only to support single and double precision types, so there is no need to allow a user of the library to implicitly instantiate the templates. That said, explicit instantiation has some issues that must be overcome. Various compilers support explicit instantiation and specialized instantiation in mutually exclusive ways. Some compilers require specialized instantiation of the static members *before* the explicit instantiation of the class (SGI Mips Pro, HP–UX, CodeWarrior 9). Other compilers require the specialized instantiation *after* the class instantiation (g++ on Linux and Macintosh, MSVC6). Yet others do not care (MSVC7.x). And the compilers sometimes cannot agree on the syntax for the instantiation. So you will find in the source files various conditional compile flags that control how the instantiation occurs. The flag that controls order of instantiation is

```
WM3_INSTANTIATE_BEFORE
```

and is defined (or not) in the file

```
MagicSoftware/WildMagic3/Source/System/Wm3Platforms.h
```

for the various platforms.

CodeWarrior 9 generates a lot of warnings about not being able to instantiate some class member functions. As it turns out, these are pure virtual member functions (no implementation required) and the warnings may be ignored.

3 Compilation of the Projects

The method for building the libraries and applications depends on which platform you are working. If you plan on installing the source code on only one platform, you need only read the subsection related to that platform.

Make certain you have satisfied the prerequisites described earlier in this document. This includes having DirectX installed (if you choose to build the DirectX renderer), having GLEW and GLUT header files copied to system include directories, and having other packages installed.

3.1 Microsoft Windows XP

The order of compilation is the same whether for MSVC6 or MSVC7.x. The only difference is the type of project file, `.dsp` for MSVC6 or `.vcproj` for MSVC7.x.

An important point that you should remember: The DirectX and OpenGL renderer projects produce the same named libraries that are copied by a post-build step to the library SDK directories. We did this to reduce the number of buildable configurations in the application projects as an attempt to simplify project management. For example, WM2 required you to rename the default configurations and create ones such as "Debug GLUT", "Debug WGL", and "Debug DX". Any 3D application had 12 different configurations

```
{Debug, DebugDLL, Release, ReleaseDLL} * {GLUT, WGL, DX}
```

The idea was for us to be able to compile all configurations for testing and support, relying on the make-like project system to compile only files that are out of date. This requires a *lot* of disk space to store all the object files, libraries, and executables (on the order of 40 GB).

In WM3, we have resorted to using just the Debug and Release configurations. The CD-ROM does not have DLL configurations. We will post these to the web site. They were not ready in time for the delivery deadline of the CD-ROM master for the *3D Game Engine Architecture* book. Because of this choice, whichever renderer project you compile last, that library is the one that will be linked into the applications. Moreover, the application library project has build configurations for GLUT, WGL, and DX. If you last built the DebugGLUT configuration for the OpenGL renderer, you need to build the DebugGLUT configuration for the application library project. If you do build different renderer configurations, we suggest doing a "rebuild all" for both the renderer and application library to make certain that no anomalies occur in linking.

We doubt that you, the application developer, will want to support more than one renderer configuration, so we believe our choice is not burdensome. If you do want to simultaneously support multiple renderers, you will have to modify the projects accordingly.

3.1.1 MSVC6 Projects

Build the projects in the following order. The path is relative to `MagicSoftware/WildMagic3`.

```
// absolutely do this one first
Source\Source.dsp

// choose one or the other
Renderers\DirectX\DirectX.dsp
Renderers\OpenGL\OpenGL.dsp

// the application library
Application\Application.dsp
```

```
// sample graphics applications (build in any order)
Test\*\*.dsp

// sample shader applications (build in any order)
SampleShaders\*\*.dsp

// sample physics applications (build in any order)
SamplePhysics\*\*.dsp

// sample image processing applications (build in any order)
SampleImagics\*\*.dsp

// sample miscellaneous applications (build in any order)
SampleMiscellaneous\*\*.dsp

// tools, some require other packages installed (see notes in Tools section)
Tools\*\*.dsp
```

3.1.2 MSVC7.x Projects

Build the projects in the following order. The path is relative to `MagicSoftware/WildMagic3`.

```
// absolutely do this one first
Source\Source.vcproj

// choose one or the other
Renderers\DirectX\DirectX.vcproj
Renderers\OpenGL\OpenGL.vcproj

// the application library
Application\Application.vcproj

// sample graphics applications (build in any order)
Test\*\*.vcproj

// sample shader applications (build in any order)
SampleShaders\*\*.vcproj

// sample physics applications (build in any order)
SamplePhysics\*\*.vcproj

// sample image processing applications (build in any order)
SampleImagics\*\*.vcproj

// sample miscellaneous applications (build in any order)
SampleMiscellaneous\*\*.vcproj
```

```
// tools, some require other packages installed (see notes in later section)
Tools\*\*.vcproj
```

3.2 Linux

From a terminal window, and assuming you are in the directory `MagicSoftware/WildMagic3`, type

```
make -f makefile.lnx
```

Sorry, but we have not yet factored out the configuration type (Debug or Release), so both are built. This takes quite some time to compile and uses a lot of disk space, on the order of 3 Gigabytes!

3.3 Macintosh

The Macintosh distribution comes with support for Apple's Xcode version 1.5 (project extension `.xcode`) and for Metrowerk's CodeWarrior 9 (project extension `.mcp`). The projects on CD-ROM have configurations only for GLUT. The projects will be modified to support both Apple OpenGL (AGL) and GLUT (as Wild Magic 2 did) and posted to the Wild Magic web site.

3.3.1 Xcode Projects

It is essential that you configure Xcode's Building settings in the following manner. From the menu bar with Xcode active, select

```
Xcode | Preferences...
```

then select the `Buildings` icon. You need to check the radio button for

```
Separate location for build products:
```

and set the path to

```
./
```

You also need to check the radio button for

```
Separate location for intermediate build files:
```

and set the path to

```
./build
```

Build the projects in the following order. The path is relative to `MagicSoftware/WildMagic3`.

```
// absolutely do this one first
Source\WildMagic3.xcode

// GLUT renderer
Renderers\OpenGL\OpenGLRenderer.xcode

// the application library
Application\Application.xcode

// sample graphics applications (build in any order)
Test\*\*.xcode

// sample shader applications (build in any order)
SampleShaders\*\*.xcode

// sample physics applications (build in any order)
SamplePhysics\*\*.xcode

// sample image processing applications (build in any order)
SampleImagics\*\*.xcode

// sample miscellaneous applications (build in any order)
SampleMiscellaneous\*\*.xcode

// tools (build in any order)
Tools\*\*.xcode
```

The main source, renderer, and application projects perform post-build steps via shell scripts, `createsdk.sh` and `fillsdk.sh`. These scripts copy the include files to a global directory

```
MagicSoftware/WildMagic3/Include
```

and the library files to global directories

```
MagicSoftware/WildMagic3/Library/{Debug,Release}
```

These directories are accessed by the applications.

3.3.2 CodeWarrior 9 Projects

These require a slight amount more work than the Xcode projects because of the lack of support for "post–build" operations. (If we are missing this support in CodeWarrior, let us know.) Launch a Terminal console window and keep it handy to run a few shell scripts that do the post-build operations.

In the terminal window, change directory to

```
MagicSoftware/WildMagic3/Source
```

Run

```
./cw_createsdk.sh
```

This creates the global Include and Library directories. (The script `createsdk.sh` is for use by Xcode; do not call this one.) The CodeWarrior project will place the compiled libraries in the appropriate directories. (The Xcode script copies compiled libraries after the build is completed.)

From a Finder window, double-click on the mcp file and build the project

```
MagicSoftware/WildMagic3/Source/WildMagic3.mcp
```

After the build is completed, from the Terminal window run

```
./cw_fillsdk.sh
```

The script copies all the source header files to the Include directory. (The script `fillsdk.sh` is for use by Xcode; do not call this one.)

In the terminal window change directory to

```
MagicSoftware/WildMagic3/Renderers/OpenGL
```

From a Finder window, double-click on the mcp file and build the project

```
MagicSoftware/WildMagic3/Renderers/OpenGL/OpenGLRenderer.mcp
```

After the build is completed, from the Terminal window run

```
./cw_fillsdk.sh
```

The script copies all the source header files to the Include directory. (The script `fillsdk.sh` is for use by Xcode; do not call this one.)

In the terminal window change directory to

```
MagicSoftware/WildMagic3/Application
```

From a Finder window, double–click on the mcp file and build the project

```
MagicSoftware/WildMagic3/Application/Application.mcp
```

After the build is completed, from the Terminal window run

```
./cw_fillsdk.sh
```

The script copies all the source header files to the Include directory. (The script `fillsdk.sh` is for use by Xcode; do not call this one.)

At this point the global Include and Library directories contain the relevant header and library files. You may build any application project at this time.

Note: CodeWarrior 9 generates a few warnings about being unable to instantiate various pure virtual class member functions. You can ignore safely these. (See the earlier section on explicit instantiation.)

4 Tools

A brief discussion of the tools is given in this section. The tools projects are found in

```
MagicSoftware/WildMagic3/Tools
```

Some of the tools are Microsoft Windows only. Each subsection mentions the platforms on which the tools will run.

4.1 3dsToWmof

Runs on Windows, Linux, Macintosh.

This is a rudimentary converter for Discreet's 3DS file format to the Wild Magic object format (wmof). The 3DS format is not very powerful. For example, it does not support skin and bones.

4.2 Bmp24ToWmif

Runs on Windows.

The program is a converter from Windows BITMAP format (bmp) to the Wild Magic image format (wmif). The usage is

```
Bmp24ToWmif myfile.bmp [myfile.alpha.bmp]
```

The input bitmaps must be 24-bit RGB. The output file is `myfile.wmif`. If you supply only `myfile.bmp`, the output is a 24-bit RGB image in the wmif format. To provide an alpha channel, you supply the second file, `myfile.alpha.bmp`, which must be 24-bit, but only the red channel is used as the alpha value. The output file is a 32-bit RGBA image in the wmif format.

4.3 CgToWm3

Runs on Windows. The Cg Toolkit is available on Linux and Macintosh, but we have not yet tested if this project automatically runs on those platforms.

4.4 ImViewer

Runs on Windows, Linux, Macintosh.

This is an image viewer for the `.im` format, a simple image format that we use for image analysis applications. The images can be of dimension two or three. If three dimensional, the up and down arrow keys allow you to change the active slice that is displayed in the window. You can use left-click-and-drag to view pixel values. The location and value are displayed in the status bar. The image types can be any native type (char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, float, or double).

4.5 Max4ToWm3

Runs on Windows. You need to have 3D Studio Max version 4 in order to compile this project. Moreover, you need to have installed the Max SDK that ships with the modeling package.

This is an exporter for 3D Studio Max. You will see empty subdirectories

```
MagicSoftware/WildMagic3/Tools/Max4ToWm3/MaxSdk
MagicSoftware/WildMagic3/Tools/Max4ToWm3/MaxSdk/include
MagicSoftware/WildMagic3/Tools/Max4ToWm3/MaxSdk/include/Maxscrpt
MagicSoftware/WildMagic3/Tools/Max4ToWm3/MaxSdk/lib
```

For convenience of access and for allowing Developer Studio's Intellisense to work on the Max SDK, we copy the SDK files from the Max installation into these subdirectories. You do not have to do this, in which case you should delete the corresponding project folders. You will also have to modify the project settings, specifically the include and library paths, to contain the paths to your Max SDK installation. You might also need to modify the output path for where the plugin DLE is written to.

4.6 Max6ToWm3

Runs on Windows. You need to have 3D Studio Max version 6 in order to compile this project. Moreover, you need to have installed the Max SDK that ships with the modeling package.

This is an exporter for 3D Studio Max. You will see empty subdirectories

```
MagicSoftware/WildMagic3/Tools/Max6ToWm3/MaxSdk
MagicSoftware/WildMagic3/Tools/Max6ToWm3/MaxSdk/include
MagicSoftware/WildMagic3/Tools/Max6ToWm3/MaxSdk/include/IGame
MagicSoftware/WildMagic3/Tools/Max6ToWm3/MaxSdk/include/Maxscrpt
MagicSoftware/WildMagic3/Tools/Max6ToWm3/MaxSdk/include/ParticleFlow
MagicSoftware/WildMagic3/Tools/Max6ToWm3/MaxSdk/lib
```

For convenience of access and for allowing Developer Studio's Intellisense to work on the Max SDK, we copy the SDK files from the Max installation into these subdirectories. You do not have to do this, in which case you should delete the corresponding project folders. You will also have to modify the project settings, specifically the include and library paths, to contain the paths to your Max SDK installation. You might also need to modify the output path for where the plugin DLE is written to.

4.7 Maya6ToWm3

Runs on Windows. You need to have Maya version 6 in order to compile this project. Moreover, you need to have installed the Maya SDK that ships with the modeling package.

This is an exporter for Maya. The project settings contain the necessary path information to locate the include and library directories of the Maya SDK. You might need to modify these for your Maya SDK installation. The output path for the plugin might also need modification.

4.8 ObjToWm3

Runs on Windows, Linux, Macintosh.

This is a rudimentary converter for Lightwave's .obj file format to the Wild Magic object format (wmof).

4.9 ScenePrinter

Runs on Windows, Linux, Macintosh.

This is a converter that processes a Wild Magic object file (wmof) and produces an ASCII representation.

4.10 SceneTree

Runs on Windows.

This is a Windows application that uses a tree control to display a Wild Magic object file (wmof).

4.11 SimpleViewer

Runs on Windows, Linux, Macintosh.

This is an application that allows you to load a Wild Magic object file (wmof) and display it. The object is centered in the view frustum. You can rotate it with the left-mouse drags (a virtual track ball is enabled). You can also move the camera around with arrow keys and other special keys.

4.12 VolumeRenderer

Runs on Windows, Linux, Macintosh.

This is an image viewer for the `.im` format, a simple image format that we use for 3D image analysis applications. The image can be rotated using left-click-and-drag mouse operations. The image types can be any native type (char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, float, or double).

4.13 WmifToBmp24

Runs on Windows.

The program is a converter from the Wild Magic image format (wmif) to the Windows BITMAP format (bmp). The usage is

```
WmifToBmp24 myfile.wmif
```

If the wmif file is RGB888, the output file is a 24-bit bmp image named myfile.bmp. If the wmif file is RGBA8888, two output files are generated. The file myfile.bmp is a 24-bit BITMAP image that contains the RGB portion of myfile.wmif. The file myfile.alpha.bmp is also a 24-bit BITMAP image, is gray scale ($R = G = B$), and contains the A portion of myfile.wmif. The converter only supports Wild Magic RGB888 and RBGA8888 for now.