WORLDTOOLKIT® RELEASE 9

WINDOWS® NT/95/98 INSTALLATION AND HARDWARE GUIDE

ENGINEERING ANIMATION, INC.
SENSE8 PRODUCT LINE
100 Shoreline Highway, Suite 282
Mill Valley, CA 94941

Telephone: (415) 339-3200
Facsimile: (415) 339-3201
Web site: *www.sense8.com*

Current version: **April 1999**

Engineering Animation, Inc.
SENSE8 Product Line
100 Shoreline Highway, Suite 282
Mill Valley, CA 94941 USA

Technical Support (e-mail): support@sense8.com
Technical Support (web site) www.sense8.com

This book was printed in the United States of America.

Contents

1

Introduction

This installation and hardware guide describes the installation, use, and special capabilities of WorldToolKit Release 9 for Windows workstations with the OpenGL graphics library.

This installation and hardware guide contains chapters on the following topics:

- System requirements and WorldToolKit (WTK) documentation sources
- Installing WTK on your Windows workstation, setting up the software license, hardware dongle configuration, and contents of the WTK distribution
- Compiling and running WTK demos and example programs as well as information concerning how WTK applications can be deployed
- Display options and notes on hardware specific to Windows workstations
- Serial Port functions
- Performance issues and specifications

See the Readme.txt file that was installed with WTK for additional installation information, last minute changes to this document or reported problems.

This chapter contains system requirements and a complete description of the WorldToolKit documentation available to you.

System Requirements

Required Hardware

To use the WorldToolKit for Windows library for development, or to run applications written using WorldToolKit, you must have a 486 or better personal computer running Windows NT 4.0, Windows 95 or Windows 98. Your system can be equipped with one or more graphics accelerator board(s) that accelerates some or all of the OpenGL graphics pipeline.

This version of WTK uses a single-processor only, although your system may have more than one processor. If you run two WTK applications at a time, Windows NT is intelligent enough to put the second WTK process on the second processor. A multi-threaded version of WTK is also available. See the section entitled Multithreaded WTK Library for more information.

WTK does not require any special graphics accelerator to run, however a graphics accelerator of some kind is strongly recommended.

WTK requires a standard PC that will run Windows NT 4.0, Windows 95 or Windows 98 with the following configuration:

- A 486/66 or better microprocessor (Pentium 90 or better strongly recommended)
- At least 24 MB of system RAM
- 85 MB of free space on the hard drive
- CD-ROM drive (for installation)

You should have at least as much free system RAM as you have dedicated texture memory on your graphics accelerator card. Because of this, your exact memory requirements depend on your WTK application. WTK keeps a copy of all textures in system memory if your machine uses any of the 3DLabs, Dynamic Pictures, and Permedia OpenGL drivers.

Also, you should be aware that OpenGL maintains a copy of all texture images loaded by your application in virtual memory. Maximize your virtual memory allocation in Windows, so it can be used to swap out images from texture memory instead of using the hard drive. Swapping to the hard drive will slow your application’s performance.

You can use WorldToolKit with just a mouse as an input device or you may want to consider one or more of the devices described in the following section.

Optional Hardware

WorldToolKit supports a wide range of 2D, 3D, and 6D input sensors, both desktop sensors and sensors worn on the body for sensing position and orientation. In addition to these sensors, you can also use devices for sound generation with WorldToolKit. The Windows version of WTK supports the following sensors:

- Any standard mouse (two button or three button)
- Ascension Bird and Ascension Extended Range Bird
- Ascension Flock of Birds
- Ascension Mouse
- CIS Graphics Geometry Ball, Jr.
- Fakespace monochrome BOOM, two-color BOOM2C, and full-color BOOM3C (button models and joystick models)
- Fakespace Pinch Glove System
- Fifth Dimension Technologies' 5DT Glove
- Gameport Joystick
- Logitech 3D Mouse (Red Baron)
- Logitech Head Tracker
- Logitech Space Control Mouse (Magellan)
- Polhemus ISOTRAK
- Polhemus ISOTRAK II
- Polhemus InsideTRAK (NT 3.51 only)
- Polhemus FASTRAK
- Polhemus Stylus
- Precision Navigation Wayfinder-VR
- Spacetec IMC Spaceball Model 2003 and Model 3003 (using only the pick button)
- Spacetec IMC Spaceball SpaceController

- StereoGraphics CrystalEyes and CrystalEyesVR LCD shutter glasses
- ThrustMaster Formula T2 Steering Console (NT only)
- ThrustMaster Serial Joystick (Mark II Flight Control/Weapons Control Systems) (NT only)
- VictorMaxx Technologies' CyberMaxx2 HMD
- Virtual i-O i-glasses! monoscopic and stereo (Intergraph only) with head tracking
- Virtual Technologies Cyberglove

Note: For Windows NT, EAI wrote special device drivers for the InsideTRAK, ThrustMaster Serial Joystick and Formula T2 Steering Console. For these sensors, the driver information needs to be entered into the registry. This is done during the installation. If related information already exists in the registry, the install program does not overwrite it and informs you about this through a pop-up window. If, however, at any time you need to manually enter these values into the registry, the specific keys and their values are discussed in the readme.txt *file in the* wtk/drivers *directory.*

Read the Readme.txt file to find out about support for devices that became available after this printing. You can construct device drivers for other devices as described in the *Writing a Sensor Driver* appendix of the WTK Reference Manual.

Optional Video Accelerators

A variety of OpenGL graphics accelerators are available from third party sources.

Required Software

WTK requires Microsoft Windows NT 4.0, Windows 98 or Windows 95 and Visual C++ 5.0 or higher.

WorldToolKit Documentation

The available sources of documentation for WorldToolKit include the following:

Installation and Hardware Guide

This installation and hardware guide tells you how to install and start WorldToolKit on computers running Windows (NT 4.0, 98 or 95) operating systems. It contains information on configuring your system for optimal performance and also contains information specific to Windows.

WorldToolKit Reference Manual

The WorldToolKit Reference Manual provides reference information and teaches you how to use WorldToolKit to create simulations.

WorldToolKit Quick Reference Guide

The WorldToolKit Quick Reference guide provides an alphabetical summary of all WorldToolKit functions, macros, and constants.The Quick Reference guide is available in PDF format (see below) on the WTK product CD; a printed version is not shipped with the product.

PDF Files

The WorldToolKit Reference Manual, the WorldToolKit Quick Reference Guide, and this installation and hardware guide are also available online in portable document format (PDF). These PDF files are automatically installed during the WTK installation process.

PDF files allow you to view and search the documents for a key word or concept. However, you must install the Adobe Acrobat 3.0 reader separately. It is included on the WTK CD-ROM or you can download it from the Adobe Acrobat web site (*www.adobe.com*).

For help within the Acrobat reader, choose Help, Reader Online Guide.

Additional Sources

See the Readme.txt file in the install directory for last minute information or reported problems. You can also find up-to-date product information by accessing EAI's SENSE8 Product Line web site at *www.sense8.com*.

2

Installation

This chapter helps you install WTK and configure the software license on your computer. The software license must be configured in order for you to compile and run WTK applications. The precompiled demo programs provided can be run even if the software license has not been configured.

This chapter contains the following sections:

- Installation information for WTK
- Software license configuration for WTK
- Hardware dongle configuration
- A description of the contents of the WTK distribution and its layout

WorldToolKit Installation

Follow these steps to install WorldToolKit on Windows NT (95 or 98) computers:

1. Exit any other applications before starting the installation process.
2. With the WorldToolKit CD-ROM in your CD-ROM drive, choose Run from the Start menu of Windows NT 4.0, 95 or 98.
3. Type `d:\setup.exe`, (where d is the letter of the drive containing the CD-ROM) and click OK.
4. When the Setup program starts, select Next at the Welcome Screen to display the Choose Destination Location screen.
5. If you want to install WTK to a directory or drive other than the default path, click the Browse button and choose a new location, then click Next.

6. At the Setup Type screen, select the type of installation you want: Typical, Compact, or Custom. Typical installs the full distribution (about 80 MB); Compact installs only the core product (about 15 MB); Custom takes you to the Custom Installation screen, where you can choose which components to install.

   Refer to the *WorldToolKit Distribution* on page 2-7 for a listing of all the directories and files installed with a Typical install. A Compact installation installs these directories: cppwrap, doc, drivers, include, lib, lightfls, makefls, and utils leaving out demo, examples, images, models, and modeler.

7. When you have selected the type of installation and the installation location, click Next on the Setup Type screen to continue.

   The Setup program tells you how much space is required to install WTK and checks for available disk space. If enough space is not available, you must choose another drive or free up some disk space and restart the WTK installation process.

8. The Setup program will begin installing the WTK distribution files on to your hard drive and then creates your program folder and program items.

9. When the files are installed, the Setup Complete screen tells you about installing the Acrobat reader, which is required to read the online documentation. You can run this installation *after* the WTK installation is complete by running d:\acrobat\setup.exe, where d is the CD-ROM drive letter. The latest version (3.0) of the Acrobat reader needs to be installed to view the online WTK documentation. The screen also tells you about the World Up demonstration program. You can run this installation *after* the WTK installation is complete by running d:\worldup\setup.exe, where d is the CD-ROM drive letter.

   The Setup Complete screen also has two checkboxes (which by default are checked). The first checkbox indicates that you would like to review the readme file now and the second checkbox causes the Volume Serial Number screen to display your system-specific volume serial number. You will need this number when you contact EAI for the WTKCODES software license code needed to run WTK.

10. Click Finish. If you checked the 'Show me my Volume Serial Number' checkbox, the Volume Serial Number screen will be displayed. Write down your system-specific volume serial number on the line below.

    System-Specific Volume Serial Number: _____

Note: You do not have to have the WTKCODES software license code to finish installing WTK. However, you must have it before running WTK applications (though you can run the precompiled demos without a WTKCODES software license code).

11. Click OK to close the Volume Serial Number screen. If you want to view this screen again, select the Volume Serial Number menu item from the WorldToolKit Program Folder.

12. If you checked the 'I would like to review the readme now' checkbox in the Setup Complete screen, the readme file will now be displayed.

Installation is now complete. To obtain and install your WTKCODES software license code, follow the directions below.

Software License Configuration

In order to compile and run a WTK based application on a computer system which has WTK installed, you must first obtain a WTKCODES software license code. (Note that the precompiled demo programs installed in the demo directory do not require a WTKCODES software license code to run.) Please contact EAI using one of the methods shown below to obtain your unique WTKCODES software license code. You must know both the serial number on the back of the CD-ROM jewel case and the system-specific volume serial number when you contact EAI to obtain a WTKCODES software license code.

Note: There are several different licensing options available for WTK. Typically, WTK is licensed to run on a specific computer via a node-locked, evaluation, or dongle license in which case the following information is applicable. (Dongle licensees would have received a hardware dongle device and should read the 'Hardware Dongle Configuration' section below.) If you have licensed WTK via a site license or a floating license, refer to the licensing instructions separately included for that type of license.

Choose one of these methods (listed in order of quickest response time) to reach EAI:

- Using your Web Browser, go to EAI's SENSE8 Product Line Web Site (*www.sense8.com*), select *Licensing*, then select *wtk codes*. Fill in the form with your name, company name, the jewel case serial number, and the system-specific volume serial number and submit the information. Your license code(s) will be e-mailed or faxed to you.

- FAX EAI at (415) 339-3201. Put "Attention: WorldToolKit Codes" as the subject. Include your name, company name, the jewel case serial number, and the system-specific volume serial number. Your license code(s) will be e-mailed or faxed to you.

When you receive the WTKCODES software license code, double-click the WTKCODES program icon to open the file. (Note that the filename of the file WTKCODES is in all capital letters and the filename has no extension.) Type the WTKCODES software license code into the file and save it. You will need the WTKCODES software license code if you reinstall the program. Write it on the line below.

WTKCODES software license code: _____

Once you've placed your system-specific software license code into your WTKCODES file, you will need to set an environment variable (WTKCODES) to the directory path where your WTKCODES file is located. If you do not set the WTKCODES environment variable, you will not be able to run WTK based applications unless they are in the same directory as the WTKCODES files. To set an environment variable in Windows NT 4.0 choose Settings from the Start menu, then select System and click the Environment tab. To set an environment variable in Windows 95 and 98, use the set command in either the autoexec.bat file or another batch file. For example, if you have installed WTK into the default installation directory (d:\Program Files\wtk), then you should set the environment variable as follows:

Variable	WTKCODES
Value	c:\Program Files\wtk

There are a number of other environment variables which are used by WTK. They are described in detail in the WTK Reference Manual's Appendix B. Especially noteworthy are the environment variables used to specify the file search path of model files (WTMODELS), texture image files (WTIMAGES), as well as the environment variable used to specify the z-buffer depth of your system (WTKZBUFFERSIZE).

Note: When making a WTKCODES *request, you can also request that EAI issue you a 64 digit S8CODES which enables the S8unlock executable to be run. The S8unlock program is used to unlock* ***non-commercial WTK applications*** *so that they can be run on computers without requiring a system-specific WTKCODES software license code. See Deploying WTK Applications on page 3-9.*

Hardware Dongle Configuration

WTK's hardware dongle license allows you to run applications on different computers without requiring a different WTKCODES license code for each computer. Contact EAI if you wish to obtain a hardware dongle.

A WTKCODES license code is required in order to use WTK with a dongle; but instead of being specific to the computer (i.e., instead of being 'node-locked'), this code is specific to the dongle you are using. So, you do not need a system-specific serial number to use a dongle. The license code (a 24 digit code), which you should have received with the dongle, must be entered into the WTKCODES file. The WTKCODES file containing this license code must be present on the computer which has the hardware dongle device connected to it.

Dongle Serial Port Issues

A dongle is a serial device. You must connect the dongle to one of the host computer's serial ports when you run your application. The dongle is equipped with a pass-through, so that you can connect another I/O device on the same port as the dongle.

Note: Devices connected to your computer through the pass-through connection on the dongle may only communicate at 9600 baud.

The serial port must have its permissions set to allow both reading and writing by the WTK application; see the Serial Ports chapter in the WTK Reference Manual for details.

By default, it is assumed that the dongle is attached to serial port 1 (COM1). If you wish to change the port, you may do so by altering the WTKCODES file. The basic entry in a WTKCODES file is a hexadecimal number (your license code). Do not alter this number. To specify the serial port you would like to use, edit the WTKCODES file so that the port or device name follows the hexadecimal number. For example, if you wish to use the dongle at serial port 2, you would alter the WTKCODES file as follows.

```
10000000xxxxxxxxyyyyyyyy   COM2
```

(Note that the addition to the file is COM2.)

Do not use quotes or other punctuation around the device name. The device name must be on the same line as the hexadecimal WTKCODES entry. You may not use ‘SERIAL1’ or ‘SERIAL2’ in the WTKCODES file. If no port is indicated after the code, the default port is assumed.

Make sure your system configuration recognizes the serial ports correctly. The configuration may have been altered so that the ports are numbered differently. [Use the Ports option in the Control Panel.]

Connecting the Dongle to Your Computer

There is only ONE correct alignment of the dongle with respect to the computer. The dongle is not reversible. The dongle has a 25 pin female interface at one end and a 25 pin male interface at the other end. ONLY the female end can be connected to a computer. The dongle should not be aligned such that the 25 pin male side faces (or, is connected to) the computer. The 25 pin male interface should either be left unused or may be used to connect another I/O device such as a joystick.

If you need extra cables, gender changers, or 25-9 pin converters, make sure they are all straight through devices. You should not be using any NULL modem connectors (or devices that interchange pin ordering) between the dongle and the host computer.

Dongle Error Messages

Couldn't open file of security codes, WTKCODES.

Cause: The WTKCODES file is missing.

Correct code not found in the WTKCODES file.

Cause: The WTKCODES entry for the dongle version of WTK is missing.

Unable to open serial port xxxx.

Cause: An invalid device name was given or permissions on the serial port are not set properly.

Dongle communication error.

Cause: The dongle is not plugged in or there may be a cabling error. Check your cabling and try again.

Dongle verification error.

Cause: The first WTKCODES entry in the WTKCODES file is not the appropriate code for your dongle. Make sure that the appropriate WTKCODES entry is at the beginning of the WTKCODES file. Contact technical support if you still have problems.

The application just hangs.

Cause: You have either not connected the dongle to the computer or the computer is unable to communicate with the dongle through the chosen port.

WorldToolKit Distribution

The installation directory (d:\Program Files\wtk by default) contains the following files and directories:

Readme.txt	Last-minute changes. *Please read this!*
WTKCODES	Text file in which to place the WTKCODES license code number for your computer. (For evaluation, node-locked, and dongle licenses only.)
wtk.ico	WorldToolKit Icon.
\cppwrap	Contains the WTK C++ wrapper library and include files which defines C++ classes for WTK's C objects so that you can use a C++ interface into WTK's functionality. A description of WTK's C++ classes can be found in the WorldToolKit Reference Manual. This directory contains 3 sub-directories (example, include, and lib) as well as a readme.txt file. NOTE: All of the components that are unique to the C++ version of WTK are contained in this directory.
\demo	Various demonstration programs and source code.
\demo\bin	Contains an executable version of the demos. These executables can be run even if the WTKCODES license code number has not been configured.
\doc	Contains Portable Document Format (PDF) versions of this Installation and Hardware Guide, the WorldToolKit Reference Manual, and the WorldToolKit Quick Reference Guide. PDF is a cross-platform file format that is read with the Adobe Acrobat 3.0 reader. You can install this Acrobat reader after the WTK

	installation is done by running d:\acrobat\setup.exe (where d is the drive letter of your CD-ROM drive). These online PDF documents are identical to the printed documents, but allow you to use a search feature to quickly find WTK functions and valuable reference information. (Tip: While you are viewing a document in the PDF reader, click the second icon on the toolbar to display the bookmarks. Then, click a bookmark to go to any chapter.) For more information on using the PDF reader, see the Adobe Acrobat help file.
\drivers	Contains calibration data for various sensor drivers.
\examples	Example programs which illustrate some of WTK's capabilities.
\images	Example BMP, JPG, RGB, and TGA (texture) files.
\include	Include files for WTK.
\lib	Contains the WTK library (wtk.lib) and the WTK UI library (wtkui.lib). In addition, there is a display list version of the WTK library (wtkdl.lib) and a multi-threaded version of the WTK library (wtkmt.lib). The display list version and the multi-threaded version of the WTK library can be used in place of the standard WTK library (wtk.lib). Refer to *Multi-threaded and Display List Versions of the WTK Library* on page 3-8 of the next chapter for more information concerning wtkdl.lib and wtkmt.lib.
\lightfls	Directory of example files containing light sources.
lights	Example file containing light sources.
\makefls	Example makefiles.
\Modeler	Contains the World Up Modeler program and Help file.
\models	Example models in the 3DS, DXF, FLT, GEO, NFF, OBJ, SLP, and WRL formats.
\redist	Contains a self-extracting executable which installs various MultiGen OpenFlight Read/Write API DLLs. WTK applications which must be able to read in OpenFlight files, must have these DLLs installed on the system where the WTK application will be run. These DLLs will be automatically installed when you install the WTK distribution. If you deploy your WTK application to other systems, you will need to run this executable on those systems if your application reads in MultiGen OpenFlight files. Note: If you force WTK to use the older OpenFlight reader available in previous

	releases of WTK (see WTuniverse_setoption) then you do not need to have the MultiGen OpenFlight Read/Write API DLLs installed on your system.
\utils	S8unlock.exe program and S8codes file for unlocking applications. This directory also contains a program called serialnum.exe, which you can use to determine your machine's system-specific volume serial number.

3

Compiling and Running WorldToolKit Programs

This chapter describes the steps involved in compiling and running WTK applications using Microsoft's Visual C++ and the procedures necessary to deploy a WTK application. A number of sample applications and example programs are provided with WorldToolKit (in the demo and examples subdirectories) to demonstrate the use of the WorldToolKit library. Precompiled versions of the demo programs are also included so that they can be run immediately after they have been installed on your system.

Using Microsoft Visual C++

The Microsoft Visual C++ Version 5.0 (or higher) compiler is required to compile WorldToolKit applications on Windows platforms. Other compilers are not supported but may work if they can handle Visual C++ generated Libraries. Before you build a WorldToolKit application, you must create a Visual C++ project workspace. (Open Visual C++ and select File>New>Project Workspace). A project workspace may be one of several types, such as, MFCAppWizard(exe), Application, Dynamic Link Library, Makefile and so on. The types that are usually used to build a WorldToolKit application are 'MFCAppWizard(exe)', 'Application' and 'Console Application'. The 'MFCAppWizard(exe)' creates C++ projects based on the Microsoft Foundation Class Library (MFC). It provides a skeleton framework with basic Windows functionality. The 'Application' is a standard Windows application to build C/C++ projects. The 'Console Application' is also used to build C/C++ projects, but it is not a Windows application, in that it consists of a console text window and a separate display window. (Applications that use neither WTK's GUI nor MFC are usually built as console applications).

Any of the above mentioned three types of workspaces can be used to build WTK applications. The process is the same for each. The application files (C/C++ source files) are inserted as the project files using Insert>Files or Project>Add Files. The include files

and the libraries to link with are specified in Build/Project>Settings (described later in the chapter). When Visual C++ successfully compiles the source code, the executable is placed in the Release (or Debug, if it is a debug application) subdirectory. You may run the application from within Visual C++ itself, or from an MS-DOS command prompt.

Some of the sample workspaces provided in the WTK distribution are:

- wtk.dsp (in the \wtk\makefls directory, where \wtk is your installation path) is an example of a 'Console Application' type of workspace, and compiles most of the demos (the non-GUI ones) that ship with WTK.
- wtkgui.dsp, also in the \wtk\makefls directory is an example of an 'Application' type of workspace, and compiles the WTK GUI applications. Note that an application that uses WTK's GUI must be a Windows application, i.e. the workspace must be built as an 'Application' type.
- wtkcpp.dsp in the \wtk\cppwrap\examples directory is a 'Console Application' type workspace that compiles applications that use WTK's C++ wrappers.
- mfcdemo.dsp in the \wtk\demo\mfcdemo directory is an example of an 'MFCAppWizard(exe)' type of workspace.

The procedures shown below, illustrate how to compile and run the wtk.c demo program using the sample workspace wtk.dsp. If you wish to compile some other source file, simply replace wtk.c with the file you wish to compile in Step 2.

1. Load wtk.dsp into Visual C++.
2. Insert the source file you want to compile. (By default, the makefile compiles wtk.c.) Use Insert>Files or Project>Add Files.

   Note that Build/Project>Settings>Additional Include Directories contains the path to the WTK include files. The libraries that this application must link with are specified in the Build/Project>Settings>Link>Input field.
3. Rebuild the project by selecting Rebuild All. This should result in the creation of an executable called wtk.exe in the makefls\console subdirectory.
4. You can run the wtk.exe program using a mouse, Spaceball, or other navigational device. To run wtk.exe using a mouse, simply select Run from the Start menu of NT 4.0 and Windows 95 or the File Manager of NT 3.51 and type the following (assuming that you have installed WTK into the default location):

"d:\Program Files\wtk\makefls\console\wtk.exe" oplan.nff

If you have a Spaceball installed on COM1, you can use it by typing:

"d:\Program Files\wtk\makefls\console\wtk.exe" -s1 oplan.nff

You should see an image of an office containing various objects. By using the mouse or Spaceball to control the viewpoint, you can fly around the office.

If you press the question mark (**?**) key when the WorldToolKit window is in focus, a list of keyboard options is displayed in the console window.

To see instructions for wtk.exe, use the '-usage' option as follows:

"C:\Program Files\wtk\makefls\console\wtk.exe -usage"

To quit, type q within the WorldToolKit window. The question mark (?), usage (-usage), and quit (q) commands are also applicable to most of the other demo programs provided in the WTK distribution.

COMMAND-LINE ARGUMENTS

You can specify command-line arguments in one of the following ways:

- Typing them at a command prompt (Start>Run)
- Assigning them to an icon on your desktop
- Assigning them in a makefile by selecting Build>Settings>Debug

Building a Project

The following steps describe how to build projects for your WTK applications.

1. Create a new workspace. (Select the type as ‘Application’ if you intend to use WTK’s GUI).
2. Include the source files to be compiled using the option Insert>Files or Project>Add Files. From Table 1 on page 3-5, determine the WTK libraries that your application will need and add them as project files.
3. Set C/C++>Code Generation>Use run-time library to:

   Debug - Debug Multithreaded DLL
   Release - Multithreaded DLL
4. Set C/C++>Preprocessor>Additional include directories to:

   \wtk\include,\wtk\cppwrap\include (and \w2w\include if you are developing multi-user applications using EAI’s World2World)
5. Add Link>Input>Object/library modules:

   The default system libraries linked into a workspace are:

   kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib

   In addition to these libraries, if you are building a standard WTK application, you will also need to link with the following libraries:

   opengl32.lib wsock32.lib winmm.lib.
6. Set Link>Ignore libraries to:

   libc.lib,libcmt.lib

Libraries, Includes, and Preprocessor Definitions

Note the following:

- All WTK libraries use the Multithreaded DLL run-time libraries.
- wtk.lib is interchangable with wtkmt.lib or wtkdl.lib.
- If you are using wtkui.lib, you must use an 'Application' style project.

Table 1 on page 3-5 lists the required libraries, includes and preprocessor definitions for each type of application you may build. The table uses the following acronyms:

- *WTK* – Standard WorldToolKit.
- *WTKUI* – WorldToolKit's user interface functions (WTui functions).
- *WTKCPP* – Standard WorldToolKit with C++ wrappers.
- *W2W* – World2World. (Requires the purchase of EAI's World2World server product.)
- *W2WCPP* – World2World with C++ wrappers. (Requires the purchase of EAI's World2World server product.)

Table 1: Libraries, Includes, and Preprocessor Definitions

Type of Application	Libraries	Includes	Preprocessor Definitions*
WTK	wtk.lib	wt.h	
WTK and WTKUI	wtk.lib, wtkui.lib	wt.h, wtkui.h	
WTKCPP	wtk.lib, wtkcpp.lib	wt.h, wtcpp.h	
WTKCPP and WTKUI	wtk.lib, wtkcpp.lib, wtkui.lib	wt.h, wtkui.h, wtkcpp.h	
WTK and W2W	wtk.lib, w2w.lib	wt.h, w2w.h	

Table 1: Libraries, Includes, and Preprocessor Definitions

Type of Application	Libraries	Includes	Preprocessor Definitions*
WTK and WTKUI and W2W	wtk.lib, wtkui.lib, w2w.lib	wt.h, wtkui.h, w2w.h	
WTKCPP and W2WCPP	wtk.lib, wtkcppw2w.lib, w2w.lib, w2wcpp.lib	wt.h, wtcpp.h, w2w.h, w2wcpp.h	W2WCPP
WTKCPP and WTKUI and W2WCPP	wtk.lib, wtkcppw2w.lib, wtkui.lib, w2w.lib, w2wcpp.lib	wt.h, wtcpp.h, wtkui.h, w2w.h, w2wcpp.h	W2WCPP

*To set preprocessor definitions: Settings>C/C++>Preprocessor>Preprocessor definitions.

MFC Issues

Retrieving MFC Events

You can build WTK applications as either console applications or MFC applications. Console-based applications receive events directly from the operating system while MFC-based WTK applications must request that events be passed to the WTK event processing system. To enable this option do the following:

```
WTuniverse_setoption(WTOPTION_USEWTPUMP, FALSE)
```

Refer to the wtk\demo\mfcdemo directory for an example of using this option. See the readme.txt file in that directory for more information.

WINDOW RENDERING AND PICKING IN MFC APPLICATIONS

When developing MFC AppWizard applications, you may need to change the way window move events and size events are handled by MFC.

Due to the way MFC handles user-interface events, picking in a moved WTK window will not work properly. In the MDI (multiple document interface) model that MFC uses, there are two components to the universe view window, the frame and the view. The user passes the view window handle to WTK for it to render into. If the user uses the MFC message pump,

```
WTuniverse_setoption(WTOPTION_USEWTPUMP, FALSE)
```

the MFC application receives all of the window messages. This isn't a problem until there is a message that WTK needs, which MFC doesn't pass on. In this case, the WM_MOVE message is sent when a window is placed somewhere on the desktop. MFC just tells the frame about the move and assumes the view could care less (unlike a resize). WTK needs to know the new location of the rendering window since the mouse data used for selection is in screen units. So, you need to subclass the standard CMDIChildWnd class used for the frame. In that class, add the following member function to handle the WM_MOVE message:

```
void WTFrame::OnMove(int x, int y)
{
    CMDIChildWnd::OnMove(x, y);
    //  See if there is a view attached to the frame
    CView* pView = GetActiveView();
    if (!pView) return;
    //  Tell the view about the move (for WTK)
    LPARAM lParam = MAKELPARAM(x, y);
    pView->SendMessage(WM_MOVE, 0, lParam);
}

void WTFrame::OnSize(UINT nType, int x, int y)
{
    CMDIChildWnd::OnSize(nType, x, y);
    //  See if there is a view attached to the frame
    CView* pView = GetActiveView();
    if (!pView) return;
```

```
        //  Tell the view about the size (for WTK)
        LPARAM lParam = MAKELPARAM(x, y);
        pView->SendMessage(WM_SIZE, 0, lParam);
    }
```

Add these functions through the class wizard.

Multi-threaded and Display List Versions of the WTK Library

A multi-threaded version of WTK is also available (\wtk\lib\wtkmt.lib). This version intelligently spawns off a second thread to break up a sequential flow of execution into two processes that can run in parallel. This simultaneous processing of different threads should improve the performance of your application. Note that to use the multi-threaded library, you DO NOT need dual processors in your machine. WTK can have multiple threads on a single processor. However, if a second processor exists, the second thread that is spawned will run on it. Refer to the benchmarks that EAI provides to gauge the difference in performance between a single-threaded version and a multi-threaded version of WTK.

A display list version of the WTK library is also available (\wtk\lib\wtkdl.lib). This version makes use of OpenGL display lists, and may give you better performance if your geometry data is not changing.

Either of these libraries (wtkmt.lib or wtkdl.lib) can be used in place of the standard WTK library (wtk.lib).

Note: The WTKCODES license required to use the multi-threaded WTK library is of a higher class than the WTKCODES license required to use the regular single-threaded WTK library. Contact EAI if you want to upgrade your license so that you can use the multi-threaded WTK library.

Deploying WTK Applications

The licensing mechanism used by WorldToolKit requires that a valid WTKCODES license code be installed on each system on which your WTK based application is to be run. Since it is sometimes necessary to demonstrate WTK based applications on non-licensed computer systems, EAI provides a tool (S8unlock) which you can use to unlock a WTK based application so that it can be run on a computer system without requiring a system specific WTKCODES license code.

The S8unlock program (located in the utils sub-directory of the WTK installation), like WTK itself, requires a system specific license code to run. You can request that a 64 digit S8CODES **non-commercial runtime** license code be issued to you to allow you to run the S8unlock program on the system you chose. The S8CODES non-commercial runtime license code issued to you allows you to unlock **non-commercial WTK based applications** so that they can be run on computers without requiring a valid system specific WTKCODES license code. WTK based applications which have been unlocked via the S8unlock program using a non-commercial runtime S8CODES license code will briefly display EAI's SENSE8 Product Line splash screen upon startup. The S8CODES non-commercial runtime license code is available to you at no extra charge.

To obtain a S8CODES non-commercial runtime license code, use your Web Browser to go to EAI's SENSE8 Product Line Web Site (*www.sense8.com*), select *Licensing*, then select *wtk codes*. Fill in the form with your name, company name, the jewel case serial number, the system-specific volume serial number, and indicate that you wish to 'obtain a S8CODES non-commercial runtime license code' in the Comments field of the form and submit the information. Your S8CODES non-commercial runtime license code will be e-mailed or faxed to you.

If you wish to deploy a **commercial WTK based application**, you will need to obtain a 64 digit S8CODES **commercial runtime** license code from EAI. A S8CODES commercial runtime license code will allow you to unlock commercial WTK based applications so that they can be run on computers without requiring a valid system specific WTKCODES license code. WTK based applications which have been unlocked via the S8unlock program using a commercial runtime S8CODES license code will not display the EAI SENSE8 Product Line splash screen upon startup. Contact your SENSE8 Product Line sales representative for S8CODES commercial runtime license code pricing.

Using The S8unlock Program

To unlock an application, type the following at the command line:

```
S8unlock <executable>
```

where <executable> is the name of an WTK application executable file. The WTK application must be in the same directory as the S8unlock program. The S8CODES license code must be placed in a file named S8codes which must also be in the same directory. (Note that the S8codes file should contain only the 64 digit license code; it should not contain any extra text, whitespace, or newlines.) If the S8CODES license code is contained in a file other than S8codes or if the file is located in some other directory, you can specify the file's location via the '-f' option. For example, if the S8CODES license code is contained in a file named 'key' in the \Program Files\my directory, you can type the following to unlock your application:

```
S8unlock -f \Program Files\my\key <executable>
```

Running S8unlock alters the executable, so it must not be write protected when you unlock it. Once you have unlocked an executable, it can be run on any computer (of the same platform) and does not require a WTKCODES license code in order to run.

The unlocked executable is modified based on the type of code found in the S8codes. The code results in an executable that is no longer node-locked to a single machine. If the code you received from EAI was for a non-commercial runtime, the executable will bring up a five second splash screen upon startup.

Additional system DLLs which may be required

WTK applications which must be able to read in OpenFlight files, must have the MultiGen Read/Write API DLLs installed on the system where the WTK application will be run. These DLLs will be automatically installed when you install the WTK distribution. If you deploy your WTK application to other systems, you will need to install the MultiGen Read/Write API DLLs on those systems if your application reads in MultiGen OpenFlight files. To install these DLLs on other systems, you can run the self-extracting executable located in the wtk/redist directory. Once you've run the self-extracting executable on a system, the MultiGen Read/Write API DLLs will be installed. Note: If you force WTK to use the older OpenFlight reader available in previous releases of WTK (see WTuniverse_setoption) then

you do not need to have the MultiGen OpenFlight Read/Write API DLLs installed on your system.

Demonstration and Example Programs

This section lists both demonstration programs and example programs. Demonstration programs are located in the wtk\demo directory (compiled versions are in the wtk\demo\bin directory). Example programs are in the wtk\examples directory.

DEMONSTRATION PROGRAMS

The WorldToolKit demonstration programs (located in the wtk\demo directory except where noted) provide useful examples that can significantly shorten the development of your own applications. This is a short summary of the demos:

arm.c	An example of using hierarchical objects (arm segments) that are connected to a base object.
belt.c	Displays a moving conveyor belt. You can drop cans onto the belt using keyboard commands.
bench.c	A demo which runs a canned set of benchmarks to measure the polygon per second performance of your system.
bounce.c	Illustrates use of the geometry constructors, dynamic Level of Detail "load management" and simple physics to create a simple scene of bouncing balls.
button.c	This demo uses 3D objects as "button" icons that follow the viewpoint as you move around the scene to construct a rudimentary example of a button-based user interface. Each button has a task associated with it.
flipobj.c	This demo resides in its own directory, \wtk\demo\flipobj. It demonstrates how to use a windows interface with WorldToolKit and uses the mouse trackball update function for object manipulation. See the Readme.txt file in this directory for more information.

fogdemo.c	This demo resides in its own directory, \wtk\demo\fog. Illustrates how fog can be used to simulate effects like fast attenuation of light, or entering a cloud.
fontdemo.c	Demonstrates using WTK 3D text font functions.
glnode	Demonstrates using WTK's OpenGL callback nodes.
gui.c	Sample WTK application using platform-independent WTK graphical user interface objects. Shows use of toolbar, slider, scrolled lists, and labels.
kitchen.c	A working model of a kitchen where the cabinet doors, drawers and oven door open or close when touched. You can turn the gas jets on the kitchen stove on, or pick up a frying pan. The models used in this demo are in the models\kitchen subdirectory.
litedemo.c	A visual demonstration of lighting capabilities. An object is loaded and placed in a "studio." Keyboard commands can create, modify, and destroy lights.
mfcdemo.c	This demo resides in its own directory, \wtk\demo\mfcdemo. It is an example of using WorldToolKit within an MFC-based application with menu and status bars and file dialogs.
missile	This demo resides in its own directory, \wtk\demo\missile. Demonstrates possible techniques for simulating fire, smoke, clouds and particle effects. Examine the README file in the missile directory for details.
morph.c	A graphical vertex manipulation demonstration. Two objects are loaded and placed left and right above a checkered floor, with a morphing object between them. The arrow keys control the transformation from the first form to the second, interpolating vertex positions and colors. The viewpoint is controlled by the mouse.
netdemo.c	If you have more than one installation of WorldToolKit connected by network hardware and software, you can run this demo, which shows how to manage a distributed simulation using WTK network calls. Up to 16 users can see representations of each other as they move around a shared world. This must be built as a Windows application, not as a console application.
papers	This demo incorporates the MathEngine SDK into a WTK application. The MathEngine SDK provides physical simulation capabilities including force, inertia, friction and elasticity modeling.

	To properly compile and run this application, you must install the SDK provided freely by MathEngine. It can be downloaded from http://www.mathengine.com
pathdemo.c	Demonstrates usage of the WTK pathing functions, creation of interpolated paths, and controlling the movement of your viewpoint or an object along the path.
pixelhit.c	This program loads in a model specified on the command line (if no model is specified, it uses rect.nff). It is designed to analyze the frequency of polygon overwrites for each pixel in the window.
portal.c	Demonstrates portalling in the latest release of WTK. Since the latest release of WTK ignores portal information (a WTK V2.1 concept) present in NFF files, this demo shows how to identify the portal polygon, make use of intersection functions and switch between two root nodes. This demo uses the models oplan.nff and lobby.nff (in the wtk/models directory), which have been updated specifically for this demo.
ripple.c	Demonstrates dynamic vertex manipulation.
shell.c	A starting point for developers. More sophisticated than simple.c, but not as comprehensive as template.c, or wtk.c.
simple.c	A basic demo showing the minimum code required for moving about a 3D environment.
sound.c	Demonstrates 3D Sound (that is, the ability to associate sound with a geometric entity).
template.c	A starting point for developers. More sophisticated than simple.c or shell.c, but not as comprehensive as wtk.c. Some basic command line arguments are recognized, and a basic keyboard handler is provided. These are to give the developer basic functionality, while leaving the rest of the development open.
texdrape.c	A graphical texture draping demonstration. The object is loaded and draped with texture in the XZ plane. Texturing and wireframe can be toggled, as well as mouse-control between the object and the viewpoint. When controlling the object, the texture is applied as stationary “slide-projector” such that it moves over the object surface. Up and down arrows scale the terrain in the y-dimension at each vertex.
treeview.exe	This demo resides in its own directory, \wtk\demo\treeview. An MFC demo that renders into multiple WTK windows (an MDI

	application). Each window has its own scene graph, so each time you load an object or a scene from a file it is brought up in a different window.
window.c	A demo showing the use of the WTK windows class and use of OpenGL drawing routines within these windows.
wtk.c	A demo which provides a general front-end to many WTK functions. It can interactively pick objects or surfaces and either apply textures, or move, modify, create, and delete objects. This demo provides examples of using many WTK functions.
wtkdiguy.c	This demo resides in its own directory, \wtk\demo\DiGuy. Demonstrates integrating Boston Dynamics, Inc.'s Di-Guy animated characters with WTK. Note: Unless you obtain the DiGuy distribution and licenses from Boston Dynamics, you will be unable to build or run this application.

If you have problems compiling the demo programs, read the Readme.txt file located in your wtk\demo directory. An executable version of each of these demos is in the wtk\demo\bin directory. You can run these demos using the program items in your WTK program group. If you have trouble running these demos, make sure you have set the environmental variables to search the correct file paths for the model and image files.Example Programs

The WorldToolKit example programs (located in the wtk\examples directory) show some useful WTK functionality. This is a short summary of the examples:

Material.c	This example allows the user to make changes to a material's property values. The default material can be used or a table can be loaded.
Ml_npath.c	Tests a motion link between a sensor and a node path. The features tested are movement in world and viewpoint frame and constraints in world frame.
Ml_path.c	Tests the usage of motion links associated with paths. Path recording, playing, and setting visibility is tested.
Ml_vpt.c	Shows different ways a sensor can control a viewpoint using motion links. Also examines the effects of adding constraints to the motion link.
Ml_xform.c	Shows sensor control of a transform node. Allows motion in different reference frames (parent and local).

Movables.c	This example demonstrates the use of movables. Movables simulate the WTK V2.1 concept of objects. They can be attached to one another to form a hierarchy, such that any transformation applied to a node higher in the hierarchy, affects all the nodes that are attached to it. So they move along with their parent node.
Nodepath.c	Demonstrates intersection testing between bounding boxes, node paths and polygons.
Pathmkr.c	Allows you to create and playback paths using WTK V2.1 and cals to the latest release of WTK.
Rv_mirror.c	Demonstrates how to achieve the effect of a rear view mirror using Viewports.
Sample.c	Use the mouse buttons to fly around a spinning planet.
Soundcre.c	Tests playing a sound, attaching the sound to a node path, starting and stopping a second sound, getting and setting parameters (in particular, gain).
Vrml.c	Sample WTK application using the platform independent WTK user interface class and instructions on how to use it with Netscape .mailcap files

4

Display Options

A variety of display options are supported by WTK. This chapter describes the hardware configuration required for each display option, as well as the appropriate display and window configuration parameters to pass in to the function WTuniverse_new.

Standard Monitor Display

No special hardware configuration is required if you simply wish to have WTK render windows on the standard monitor for viewing on-screen. If you pass in WTDISPLAY_DEFAULT as the first argument to WTuniverse_new, WTK puts up a single color display window when the second parameter is WTWINDOW_DEFAULT or a stereo pair of such windows when the second parameter is WTWINDOW_STEREO. You can then use the function call WTwindow_new to create additional WTK windows.

Stereoscopic Displays

Modes of Stereoscopic Viewing

Refer to the WTK Reference Manual's Universe chapter for information about stereoscopic viewing.

Using Head-mounted Displays

To generate true-color stereoscopic imagery suitable for use with a head-mounted display (HMD), two channels of video output are required. On the Intergraph platform this requires two graphics boards configured to run in dual mode. On other systems, consult your hardware documentation to see if two boards can be used in a single system.

If you plan on using a head-mounted display (HMD), you should be aware that they require different types of video formats. Please contact your HMD supplier to learn which method works best with their HMD.

The common HMD video formats are:

- **VGA**, usually 60Hz, a 640x480 RGB non-interlaced signal with separate sync signals (H,V).
- **NTSC composite**, (just like the video-in RCA jack on your VCR). If you intend to use an NTSC-based head mounted display, you can either purchase a VGA to NTSC scan converter or use a graphics board (like Intergraph's), which allow you to output a RS-170 signal. You can then use an NTSC encoder to convert this to NTSC composite.
- **S-video,** is a variation on the NTSC composite signal with improved color support.
- **RS-170**, is an interlaced RGB signal with NTSC timings.
- **RGB**, usually 60Hz, resolutions from 640x480 up to 1024x1280.
- **RGB frame sequential**, is a very special format that requires running the video subsystem at 180Hz and generating each component separately at 60Hz (for a total of 180Hz). It's best to consult your HMD manufacturer about this mode. Some HMD's require 180 Hz field-sequential RGB (e.g., Virtual Research FS5, NVision). Intergraph provides a special driver for these devices.

If you plan on using your HMD in stereo mode, most HMDs require two independent video signals to do this. You will have to examine your graphics hardware to see if it is capable of generating multiple video signals.

See also the discussion in the *Viewpoints* chapter of the WTK Reference Manual about the viewpoint convergence value, used to fuse the images coming into the HMD. If this value is large, it may be preferable to create your own stereo pair of windows as described there, rather than to use the WTWINDOW_STEREO option in WTuniverse_new.

Stereoscopic Devices

StereoGraphics' CrystalEyes

Stereo images can be displayed using StereoGraphics Corporation's CrystalEyes LCD shutter glasses. With this display option, a high-resolution stereo image is created that multiple viewers can view through inexpensive and lightweight LCD shutter glasses.

In order to use CrystalEyes with WorldToolKit, you need the following devices:

- StereoGraphics CrystalEyesPC infrared sync signal generator.
- StereoGraphics CrystalEyes LCD shutter glasses and/or head tracking.
- Stereo-ready monitor. This means a monitor capable of accepting a 120Hz video signal. Please contact Intergraph or StereoGraphics directly for the list of monitors currently available.

Refer to the Sensors Setup Guide on the technical support page of EAI's SENSE8 Product Line Web Site (*www.sense8.com*) for detailed information on how to configure the above devices. After hooking up these devices per the StereoGraphics instructions, you will then need to pass in WTDISPLAY_CRYSTALEYES to the WTuniverse_new function to tell WorldToolKit to display left and right stereo images on a single display. This is the 'Over/Under' mode of stereoscopic display.

Note that this display option creates a full-screen stereo display. On some monitors, it may be useful to use the WTscreen_setyblank call to adjust the vertical blanking interval, that is, the vertical space between the left and right eye video signals. This can correct vertical alignment problems between the left and right eye images.

EAI added CrystalEyes support to the standard wtk.exe demo so that you can see how this works. After hooking up all the hardware and video signals, start the program by typing the following at the command prompt:

```
\wtk\demo\wtk.exe -c oplan.nff
```

where, \wtk is your installation path. (You should have the environment variable WTMODELS set to point to the models directory for the program to locate oplan.nff).

You should see both images overlapped on your 120Hz monitor without the LCD glasses. If the two images are exactly overlapping, you will want to modify your parallax setting to a non-zero value. If you don't, you won't see a 3D image when you put your LCD glasses on.

Turn on your CrystalEyes LCD shutter glasses and put them on. You may need to modify the convergence distance and parallax values until the desired stereo effect is achieved. By experimenting with these two parameters, you will learn how they can be used to exaggerate or diminish the stereo effect, make objects appear to be behind or in front of the screen, and/or eliminate bothersome ghosting effects.

CrystalEyesVR

Another product from StereoGraphics, called CrystalEyesVR, incorporates a Logitech ultrasonic head-tracker into the LCD glasses. By moving your head from side to side or up and down, you can see around to either side or above and below an object. The physical set up of the CrystalEyesVR head-tracker is shown in your WTK Reference Manual in the section that describes the CrystalEyesVR sensor functions.

To use CrystalEyesVR, configure your display as described above for CrystalEyes. In addition, create a CrystalEyesVR sensor object as described in the WTK Reference Manual and attach it to the viewpoint via a motion link.

Virtual i-O i-glasses!

Another option for displaying stereo images is with Virtual i-O's i-glasses! With this display option, a 640x480-resolution stereo image is created for a single user.

In order to use the glasses with WorldToolKit, you need to connect the glasses to your Intergraph GLZ graphics boardset. Refer to the Sensors Setup Guide on the technical support page of EAI's SENSE8 Product Line Web Site (*www.sense8.com*) for detailed information on how to configure this device. Be sure to set the video resolution to 640x480 and reboot your workstation.

After following these steps, you will then need to pass in WTWINDOW_STEREO to the WTuniverse_new function to tell WorldToolKit to display using OpenGL left and right stereo buffers on a single display.

EAI added stereo in a window support to the standard wtk.c demo so that you can see how all this works. After hooking up all the hardware and video signals, start the program by typing the following at the command prompt:

```
\wtk\demo\wtk.exe -k oplan.nff
```

where, \wtk is your installation path. (You should have the environment variable WTMODELS set to point to the models directory for the program to locate oplan.nff).

You should see both images overlapped on your 640x480 monitor without the glasses. If the two images are exactly overlapping, modify your parallax setting to a non-zero value. If you don't, you won't see a 3D image when you put on your glasses.

Turn on your glasses and set them in stereo mode. You may need to modify the convergence distance and parallax values until the desired stereo effect is achieved. By experimenting with these two parameters, you will learn how they can be used to exaggerate or diminish the stereo effect and make objects appear to be behind or in front of the screen.

5

Serial Port Functions

These functions are specific to Windows platforms only. For more information on serial port functions, see the *Serial Ports* chapter of the WTK Reference Manual.

WTserial_new

```
void WTserial_new(
    char *port,
    int baud,
    char parity,
    int databits,
    int stopbits,
    int buffersize);
```

WTserial_new creates and returns a new WorldToolKit serial port object. The first argument is the device filename, for example, SERIAL1 for serial port 1. The baud rate must be one of 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, or 115200. Parity is E, M, O, or N for Even, Mark, Odd or None. Databits is the number of bits per byte which must be any of 4, 5, 6, 7 or 8. Stopbits is the number of stop bits per byte which must be either 1 or 2. Buffersize specifies the size of the input and output communications buffers for the device. See the include file sensor.h for an example of this function's usage.

This function will not work if you have deleted all the COM ports from your Windows desktop (see your Windows documentation to add a COM port). When you run a WTK application that uses WTserial_new, the Windows COM port settings (like baud rate) are temporarily overwritten by the settings selected for WTserial_new. When you close your WTK application, the Windows COM port settings are returned to their original values.

WTserial_setRTS

```
FLAG WTserial_setRTS(
    WTserial * serial,
    FLAG set);
```

WTserial_setRTS allows the RTS signal to be enabled or disabled for the device pointed to by serial. Some platforms and devices require the signal to be set either high or low before serial communication with a device is possible. Pass in TRUE for the set argument to set the RTS line high, or FALSE to set the RTS line low. WTserial_setRTS returns TRUE if successful, otherwise FALSE is returned.

WTserial_setbytesize

```
void WTserial_setbytesize(
    WTserial *serial,
    int size);
```

Use this function to set the number of bits per byte. This information is used for communication over the serial port specified by “serial.” The requested size is passed in as “size.” The function returns TRUE if it succeeds in setting the value or FALSE otherwise.

WTserial_getbytesize

```
int WTserial_getbytesize(
    WTserial *serial);
```

This function returns the number of bits per byte for the serial object pointed to by serial.

WTserial_setbaud

```
FLAG WTserial_setbaud(
    WTserial * serial,
    int baud);
```

WTserial_setbaud sets the baud rate for the serial object pointed to by serial. The values for baud can be any of the following: 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, or 115200. WTserial_setbaud returns TRUE if the action succeeded, otherwise FALSE is returned.

Note: Some devices require the baud rate to change after opening the devices.

WTserial_getbaud

```
int WTserial_getbaud(
    WTserial * serial);
```

WTserial_getbaud returns the current baud rate for the serial object pointed to by serial.

6

Performance Issues

Each platform and OpenGL graphics accelerator has different performance characteristics. Each system has different abilities to render objects with specific geometries, textures, and lighting, depending on these factors:

- The graphics hardware accelerator installed
- The amount of texture memory and main memory in your system
- Other aspects of your system configuration including the level of optimization of OpenGL on your platform

You can use the benchmark program bench.exe (located in the \wtk\demo\bin directory) to help identify performance issues with your system.

This chapter contains performance specifications and some guidelines and tips for optimizing performance using WTK. Some experimentation is typically necessary to optimize the frame rate on your particular system for a given application.

Performance Specifications

This is a tricky subject, and you will hear many different kinds of numbers mentioned by different manufacturers. Our numbers represent the performance achievable in a user's application rather than raw hardware numbers.

Use the benchmark program, \wtk\demo\bin\bench.exe, to directly measure how fast WTK is on your system. It can measure the fill rate, transform rate, and general performance of your CPU.

It is important to note that the frame rate of an unchanging scene (on a 60Hz display) may be quantized at 60 fps, 30 fps, 20 fps, 15 fps, 12 fps, etc. because of the sync to top-of-frame which happens when the image buffer is swapped in the double-buffering process. On a

72Hz display, the frame rate is quantized at 72 fps, 36 fps, 24 fps, 18 fps, 14.4 fps, etc. What this means is that if you have an application (e.g., the benchmark program) running at 30Hz, it is possible that making a very minor change in the application (like removing a few polygons from the scene) will double your frame rate, i.e., jump to 60Hz! The performance figures below are benchmark figures that were obtained using the \wtk\demo\bin\bench.exe program.

The reported rates are the highest rates achieved by adjusting the parameters to this benchmark program; the parameters used are given in each case. This yields essentially the highest polygon per second rates achievable in a WTK application for the given platform. If you get lower results, try adjusting the parameters to bench.exe downward a little; the parameters specified are (in all cases) very close to a quantization border, and if you are on the "wrong side" of that border, your frame rate will be cut in half.

Table 6-1 below is for a 120MHz Pentium processor system running NT 3.51 with Intergraph's Realizm Z13 board and rendering into a 640x480 window. The display was 1024x768 resolution running at 60Hz at 24-bit color.

Table 6-1: Intergraph Realizm Z13 performance

Test	Non pre-build	Pre-build
40x40 test, Gouraud (smooth) shaded *lit, Z-buffered, 96-pixel, triangle mesh*	19,138* Triangles/sec	53,872* Triangles/sec
40x40 test, Textured *lit, Z-buffered, 96-pixel, triangle mesh, trilinear mipmapped, Gouraud-shaded, perspective correct*	19,500* Triangles/sec	43,596* Triangles/sec
Gouraud-shaded fill rate *lit, Z-buffered, front-to-back*	65 Megapixels/sec	65 Megapixels/sec
Textured fill rate *trilinear mipmapped, Gouraud-shaded, lit, Z-buffered, front-to-back*	33 Megapixels/sec	33 Megapixels/sec
Terrain measurement *trilinear mipmapped, Gouraud-shaded, lit, Z-buffered*	4.7* Frames/sec	7.7* Frames/sec

** These measurements were limited primarily by the speed of the CPU, not the graphics board*

Note the significant difference between using WorldToolKit's pre-built (optimized) mode and the default mode for most of the measurements. In general, you want to render all your objects in this mode, see WTgeometry_prebuild in the WTK Reference Manual for more information. You should also be aware that when it comes to rendering lots of polygons, the CPU usually becomes the bottleneck long before the graphics board.

Table 6-2 below is for a 120MHz Pentium processor system running NT 3.51 with Intergraph's Intense3D board and rendering into a 640x480 window. The display was 1024x768 resolution running at 60Hz at 24-bit color.

Table 6-2: Intergraph Intense3D performance

Test	Non pre-build	Pre-build
40x40 test, Gouraud (smooth) shaded *lit, Z-buffered, 96-pixel, triangle mesh*	19,323* Triangles/sec	49,922* Triangles/sec
40x40 test, Textured *lit, Z-buffered, 96-pixel, triangle mesh, trilinear mipmapped, Gouraud-shaded, perspective correct*	19,138* Triangles/sec	49,922* Triangles/sec
Gouraud-shaded fill rate *lit, Z-buffered, front-to-back*	29 Megapixels/sec	29 Megapixels/sec
Textured fill rate *trilinear mipmapped, Gouraud-shaded, lit, Z-buffered, front-to-back*	12 Megapixels/sec	12 Megapixels/sec
Terrain measurement *trilinear mipmapped, Gouraud-shaded, lit, Z-buffered*	4.5* Frames/sec	6.9* Frames/sec

** These measurements were limited primarily by the speed of the CPU, not the graphics board*

You can do a lot within an application to optimize performance. A number of tips for optimizing performance with the NT version of WTK are provided in the remaining sections of this chapter.

Improving Performance

- *Use just one WTK light to light the scene.* Lights are expensive, in terms of performance. Spot lights slow performance the most, followed by point lights, then directed lights. Ambient lights have the least effect.
- *Use WorldToolKit's prebuild option.* For some geometry, this will more than double your performance. See WTgeometry_prebuild for more details.
- *Use a visibility table.* Quick reject of objects does not work well for objects with large spatial extent. There is a trade-off, however, between the overhead of managing many small objects and the use of small objects for quick reject. You may have to do some experimentation to discover what works best for your application. Impressive performance gains have been obtained, particularly for architectural models, by segmenting large worlds into smaller objects, not all of which are visible from each other. By creating a visibility table of the objects that can be seen from within other objects, object visibilities can be appropriately turned on and off.
- *Render only what is visible*. Use WTnode_enable to prevent distant or otherwise obscured objects (and the children of those objects) from being rendered.
- *Use simpler models when possible*. For example, when objects will be viewed at a distance at which detail would not be visible, or when detail can be replaced by the use of textures (see below). You can use Level of Detail nodes to render simplified versions of your models which are swapped in at distances where detail is less important.
- *Use the default cursor*. Some systems, Intergraph for example, render slower if cursors other than the Windows default cursor are used. The windows default cursors are drawn in hardware on these systems but the other cursors are drawn in software.
- *Use scene graphs*. Structure your environment as different scene graphs, limiting the number of polygons in each scene graph.
- *Make sure you have enough system memory*. Just for texture storage, you need as much system memory as your application uses for all loaded textures. The OpenGL accelerator will perform paging of textures into or out of on-board texture memory, but a copy of the texture is always maintained by OpenGL in memory — you don't want to page to the hard disk if you can avoid it.

- *Make single-sided models.* Create your models such that as many faces as possible are backface-rejected. This means fine-tuning your models to eliminate surfaces that are viewable from both the front side and the back side. Many times only a single direction needs to be viewable. Use the World Up Modeler to fine-tune your geometry.
- *Avoid excessive use of collision detection*. Try to do this only where critical and do it intelligently so that only objects that could theoretically collide are tested.
- *Merge coplanar polygons*. If you have a flat surface that has 100 polygons in it (a 10x10 mesh for example), you can convert this to a single polygon as long as all the vertex properties are shared by all the polygons. The World Up Modeler will do this for you. This can eliminate a lot of polygons without any change in appearance.
- *Scale the model in the modeling program*. Do not use too large of a scale factor when loading models with WTnode_load. Using a scale factor which is too large (typically more than about 100.0) will cause WorldToolKit's renderer to slow down due to issues related to arithmetic precision. Perform any necessary scaling in your modeling program and use a scale factor of 1.0 when loading the geometry into WorldToolKit.
- *Test with simpler models*. Instead of moving a complex object, try moving either a bounding box or a simplified version of the object. When movement is complete, swap the complex object back in.
- *Use square partitions for databases.* When partitioning a database, create square partitions rather than rectangular ones. The culling of partitions is based on the radius of the bounding box surrounding the object.
- *Use shaded and transparent textures sparingly.* Shaded and transparent textures render more slowly than ordinary textures. Drawing a polygon with transparent textures is perhaps 20 percent slower than drawing the same polygon with a flat-shaded texture.
- *Avoid texture swapping.* Do not exceed your system's texture memory.
- *Use perspective textures sparingly if you don't have a hardware accelerator.*

- *Use textures to reduce the polygon count*. Use textured surfaces in place of modeled detail. Although textured polygons render more slowly than untextured polygons, the savings in total number of polygons in the model may be substantial when textures are used. Assume it takes twice as long to draw a textured polygon as a non-textured one, so use this trade-off appropriately. You can create very realistic scenes, for example, by using just one or two transparently textured polygons to represent a tree, or four textured rectangles to form the exterior walls of a building.
- *Tune your application*. Several WorldToolKit functions and demo programs may help you tune your application for maximum performance.

  Functions include:

  - WTgeometry_numpolys, which returns a count of the number of polygons present in the specified geometry.
  - WTuniverse_framerate, which returns the number of frames per second at which the application is running.
  - WTtexture_getmemory, which determines how much texture memory is being used by your application.
  - WTwindow_numpolys, which tells you how many polygons have been drawn this frame.
  - WTgeometry_prebuild, converts geometry into triangle or quad strips for faster performance.

  Demo programs include:

  - pixelhit, which will read your geometry and output a map of the number of pixel overwrites from the viewpoint. This is important information when analyzing performance. The goal is to achieve less than 1.5 pixel overwrites per frame.
  - bench, a tool for examining many different aspects of performance.
  - The World Up Modeler, which allows you to flip the direction of polygon faces, turn bothsides on or off, merge coplanar polygons and turn vertex normals on or off.

Enhancing Intergraph Performance

The following features are unaccelerated on the GLZ1T, GLZ5, GLZ6, and GLI boardsets:

- Texturing (any primitive) with any of the following attributes:
    - Clamp mode
    - 4 component DECAL
    - 1 and 2 component BLEND
    - Textures with borders
- Polygons (triangles, triangle strips, quads, quad strips) with anti-aliasing enabled
- Lines (lines, line strips) with texturing and anti-aliasing both enabled

Index

V

W