User Guide

GLExpert

NVIDIA Performance Toolkit

DEVELOPMENT

Table of Contents

Introduction

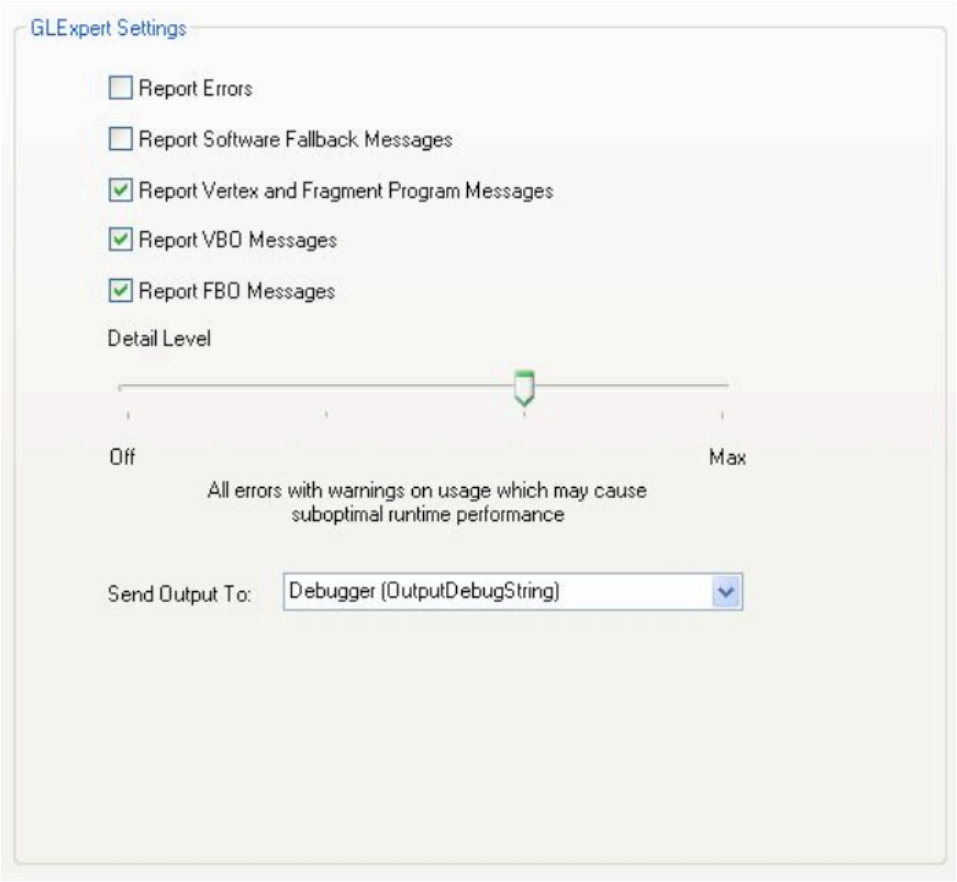
GLExpert is a part of the instrumented driver provided with NVPerfKit 2.0. It provides OpenGL application developers with real-time debugging information from the OpenGL runtime to help track down API usage errors and performance issues. Developers are able to choose specific categories of interest, as well as the level of information detail they wish to receive, using either the NVIDIA Developer Control Panel (NVDevCPL), or a programmatic interface provided via NVAPI.

System Requirements

- NVIDIA instrumented display driver, version 84.15 or later on Windows XP
- NVPerfKit including the Developer Control Panel (NVDevCPL)
- NVAPI (see accompanying documentation) library and header file, if using the programmatic interface

GLExpert Configuration

GLExpert can be configured two ways: using the new GLExpert pane in the NVDevCPL provided with NVPerfKit 2.0 (pictured below), or using a programmatic interface provided via NVAPI.



Note: GLExpert, when configured with the NVDevCPL, is system-wide and impacts all OpenGL applications. However, any OpenGL applications that are running when the settings are applied will need to be restarted for the changes to take effect.

GLExpert can also be configured using a programmatic interface provided via NVAPI (see accompanying documentation). Through this API, the user can configure the same properties they can through the NVDevCPL (all of which are system-wide), with two important additions. First, you can apply settings to the only current process and have those settings revert to the system defaults when the process quits (useful for in-engine, dynamic configuration of GLExpert). Additionally, you can register a callback from your engine, which GLExpert will call for each message generated by the runtime. This allows for full flexibility in processing, filtering, or debugging using GLExpert.

There are three configuration options that developers will set, no matter which interface they choose to use: the report categories of interest, the level of information detail, and finally, the output location.

Categories

The developer first determines which areas of OpenGL functionality they are interested in investigating; these are referred to as report categories. Available options include OpenGL Runtime Errors and messages concerning VBO or FBO usage and performance. The developer may select as many of these as they want, as long as at least one category is selected. If no categories are selected, GLExpert output is effectively disabled.

Level of Information Detail

This setting allows the developer to determine the level of information detail they wish to receive. The slider has several positions, and as it is moved toward Max, the amount and nature of the information changes. A brief description appears beneath the slider as it is moved. As the slider progresses to the right, the settings are additive, so all messages from lower settings are included in higher settings.

Note: With the slider set to Off, GLExpert output is effectively disabled. All other slider positions are referred to as "active" settings.

At the lowest active setting, GLExpert will report serious issues such as OpenGL Runtime Errors (if the proper category has been enabled) or usage that may affect rendering correctness. As the slider is moved toward the Max setting, additional warnings may be reported, generally concerning performance or unusual, possibly non-performant, usage patterns. Finally, at the Max setting, detailed usage information and some resource tracking statistics will be included in the output message stream.

Output Location

GLExpert output can be sent to either the console (via standard out), to a debugger (via OutputDebugString), or to a user-specific callback function. The message sent to all locations is identical. However, as detailed below, there are minor differences to the packaging of the message for delivery.

GLExpert Message Format

When GLExpert sends a message to either the console or debugger, it prefixes the message with a single "`OGLE:`" to allow for simple parsing, followed by a pair of IDs (described below). These same IDs are passed as parameters to the user-specific callback, if one is configured and that output mode is enabled. The message itself makes up the remainder of the output, and is identical for all output modes.

A GLExpert message is composed of two parts: a base message and a context-specific message. There is no separator in the stream between these two parts, as the two together make up a complete message.

The base portion of the message gives general information on the nature of the message (correctness, performance, information). For example:

```
The current FBO state (e.g. attachments, texture targets)
is UNSUPPORTED.
```

The context-specific portion of the message details specifics about the situation in which this message is being reported (the object name, the error code, the particular observed usage pattern). For example:

```
Reason: COLOR_ATTACHMENT0 attempting to bind to an
unsupported texture target.
```

Each message is also accompanied by two IDs: one specifies which category the message belongs to (category ID) and the other is unique to that base message (message ID). The category ID is consistent across all messages from a single category, but the message ID is unique only for messages issued with a common base message (as described above); the context-specific message will vary depending on the runtime state generating the message, but will use the same message ID.

Combining all of the formatting produces a GLExpert message:

```
OGLE: Category: 0x00000010 MessageID: 0x00840000

The current FBO state (e.g. attachments, texture targets)
is UNSUPPORTED. Reason: COLOR_ATTACHMENT0 attempting to
bind to an unsupported texture target.
```

Appendix A.
GLExpert Base Message Reference

This reference includes all of the currently supported base messages, along with their category ID, message ID, and base message text.

NVAPI_OGLEXPERT_REPORT_ERROR
Category ID: 0x00000001

The ERROR category controls the reporting of OpenGL errors as they occur.

Message ID : **0x00800000**
Message Name : ERROR_GENERAL
Message Level : 10

An OpenGL error has occurred

Message ID : **0x00800001**
Message Name : ERROR_INVALID_ENUM
Message Level : 10

A provided enum value is out of range

Message ID : **0x00800002**
Message Name : ERROR_INVALID_VALUE
Message Level : 10

A provided numeric argument is out of range

Message ID : **0x00800003**
Message Name : ERROR_INVALID_OPERATION
Message Level : 10

A provided numeric argument is out of range

Message ID : **0x00800004**
Message Name : ERROR_STACK_OVERFLOW
Message Level : 10

The current operation would cause a stack overflow

Message ID : **0x00800005**
Message Name : ERROR_STACK_UNDERFLOW
Message Level : 10

The current operation would cause a stack underflow

Message ID : **0x00800006**
Message Name : ERROR_OUT_OF_MEMORY
Message Level : 10

Not enough memory left to execute the current operation

Message ID : **0x00800007**
Message Name : ERROR_TABLE_TOO_LARGE
Message Level : 10

The table specified is too large

NVAPI_OGLEXPERT_REPORT_SWFALLBACK
Category ID : 0x00000002

The SWFALLBACK category controls the reporting of situations where the driver has fallen back to software for at least part of the graphics pipeline. This has obvious performance impacts, and should always be avoided.

Message ID : **0x00810000**
Message Name : SWFALLBACK_GENERAL
Message Level : 20

Software rendering has been enabled

Message ID : **0x00810001**
Message Name : SWFALLBACK_FORCED
Message Level : 20

Software rendering has been enabled by default

Message ID : **0x00810002**
Message Name : SWFALLBACK_RENDER_MODE
Message Level : 20

Falling back to software because RenderMode != GL_RENDER

Message ID : **0x00810003**
Message Name : SWFALLBACK_UNSUPPORTED_VERTEX
Message Level : 20

Falling back to software because a transform-related option is not supported with the current hardware configuration

Message ID : **0x00810004**
Message Name : SWFALLBACK_UNSUPPORTED_PROGRAM
Message Level : 20

Falling back to software because a program-related option is not supported with the current hardware configuration

Message ID : **0x00810005**
Message Name : SWFALLBACK_UNSUPPORTED_DEPTH_STENCIL
Message Level : 20

Falling back to software because the specified depth or stencil operation is not supported with the current buffer and hardware configuration

Message ID : **0x00810006**
Message Name : SWFALLBACK_UNSUPPORTED_TEX
Message Level : 20

Falling back to software because a texture object is using an unsupported format

Message ID : **0x00810007**
Message Name : SWFALLBACK_UNSUPPORTED_ROP
Message Level : 20

Falling back to software because the specified raster operation is not supported with the current buffer and hardware configuration

Message ID : **0x00810008**
Message Name : SWFALLBACK_NONACCELERATED_RESOURCES
Message Level : 20

One or more render buffers are not accelerated likely as a result of other fallbacks

Message ID : **0x00810009**
Message Name : SWFALLBACK_UNSUPPORTED_EMULATION_REQUIRED
Message Level : 20

Falling back to software because emulation is required for rendering

NVAPI_OGLEXPERT_REPORT_PROGRAM
Category ID : 0x00000004

The PROGRAM category deals with information pertaining to shaders and programs, high-level or assembly, including compiling and linking, usage, and performance.

Message ID : **0x00820000**
Message Name : PROGRAM_COMPILE_FAILED
Message Level : 10

The provided shader has failed to compile

Message ID : **0x00820001**
Message Name : PROGRAM_LINK_FAILED
Message Level : 10

The provided program has failed to link

NVAPI_OGLEXPERT_REPORT_VBO
Category ID : 0x00000008

The VBO category controls the reporting of VBO-related events and behaviors. This includes situations such as when the observed usage pattern could result in sub-optimal performace or appears contradictory to user-provided usage hints, when buffer objects are moved from one memory space to another (i.e. promotion or demotion), when (and which) buffer objects are used by the GPU, or when buffers are updated with new data.

Message ID : **0x00830000**
Message Name : VBO_WARNING_USAGE
Message Level : 10

The current VBO usage pattern may not be what was intended

Message ID : **0x00830001**
Message Name : VBO_WARNING_PERFORMANCE
Message Level : 20

The current VBO usage pattern may result in unexpected performance losses

Message ID : **0x00830002**
Message Name : VBO_WARNING_MOVEMENT
Message Level : 20

A buffer object has been moved (copied) to another memory space

Message ID : **0x00830003**
Message Name : VBO_INFO_MAPPING
Message Level : 30

A buffer object has been mapped

Message ID : **0x00830004**
Message Name : VBO_INFO_UPDATE
Message Level : 30

A buffer object has been updated and the changes are being propagated

Message ID : **0x00830005**
Message Name : VBO_INFO_CONFIG
Message Level : 30

A buffer object has been configured for use (e.g. memory space selected)

```
NVAPI_OGLEXPERT_REPORT_FBO
Category ID : 0x00000010

The FBO category controls the reporting on FBO-related events and
behaviors. This includes any errors or warnings that can occur
during binding, completeness checks, or usage.

Message ID    : 0x00840000
Message Name  : FBO_UNSUPPORTED
Message Level : 10

The current FBO state (e.g. attachments, texture targets) is
UNSUPPORTED

Message ID    : 0x00840001
Message Name  : FBO_UNSUPPORTED_OUT_OF_MEMORY
Message Level : 10

The current FBO operation has run out of memory and cannot
complete
```

Contact

Please let us know if you encounter any problems or think of additional features that would improve NVPerfKit. You can reach us at the following email address:

NVPerfKit@nvidia.com

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com