



GPU Programming Guide GeForce 8 and 9 Series

December 19, 2008

0001

Exh. 2001
IPR2015-00326
U.S. Pat. No. 6,897,871

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, and NVIDIA Quadro are registered trademarks of NVIDIA Corporation. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2008 by NVIDIA Corporation. All rights reserved.

HISTORY OF MAJOR REVISIONS

Version	Date	Changes
1.0	12/19/2008	Initial

Table of Contents

Chapter 1. About This Document	7
1.1. Introduction	7
Chapter 2. How to Optimize Your Application	9
2.1. Making Accurate Measurements	9
2.2. Finding the Bottleneck	10
2.2.1. Understanding Bottlenecks	10
2.2.2. Basic Tests	12
2.2.3. Using PerfHUD	12
2.3. Bottleneck: CPU	13
2.4. Bottleneck: GPU	15
Chapter 3. General GPU Performance Tips	17
3.1. List of Tips	17
3.2. Graphics API Overhead (CPU)	19
3.2.1. Use Fewer Batches	19
3.2.2. Reduce state changes and constant changes	20
3.3. Vertex Processing	20
3.3.1. Use Indexed Primitive Calls	20
3.3.2. Attribute bottleneck (Vertex Setup)	21
3.4. Shaders	21
3.4.1. Choosing the latest shader model	21
3.4.2. Compile with the latest compiler available.	22
3.4.3. Choose the Lowest Data Precision That Works	22
3.4.4. Save Computations by Using Algebra	23
3.4.5. Don't Write Overly Generic Library Functions	24
3.4.6. Don't Compute the Length of Normalized Vectors	24
3.4.7. Fold Uniform Constant Expressions	24
3.4.8. Uniform Parameters Caveat	25
3.4.9. Pixel Shader Bottlenecks?	25
3.4.10. Interpolants and Post-transform cache pressure.	26
3.4.11. Geometry Shaders?	27

3.4.12.	Use the <code>mul()</code> Standard Library Function	27
3.4.13.	Use <code>D3DADDRESS_CLAMP</code> (or <code>GL_CLAMP_TO_EDGE</code>)	28
3.4.14.	Too many generated primitives in Geometry Shader	28
3.5.	Texturing	28
3.5.1.	Use Mipmapping	28
3.5.2.	Use Trilinear and Anisotropic Filtering Prudently	29
3.5.3.	Replace Complex Functions with Texture Lookups	29
3.6.	Rasterization	30
3.6.1.	Double-Speed Z-Only and Stencil Rendering	30
3.6.2.	Z-cull Optimization	31
3.6.3.	Lay Down Depth First ("Z-only rendering")	31
3.6.4.	Allocating Memory	31
3.7.	Antialiasing	32
3.7.1.	Coverage Sampled Anti-Aliasing (CSAA)	33
Chapter 4. GeForce 8 and 9 Series Programming Tips		34
4.1.	Introduction to GeForce 8/9 series architecture	34
4.2.	Shader Model 4.0	35
4.3.	Shader Model 4 System Values (SV_)	36
4.3.1.	System Values Performance Tips	36
4.4.	Vertex Setup/Attribute bottleneck issues	37
4.4.1.	Vertex assembly on a GPU	37
4.4.2.	Attribute bottleneck	37
4.4.3.	Detecting Attribute bottlenecks	38
4.4.4.	Fixing Attribute bottlenecks	38
4.5.	Vertex Texture Fetch	40
4.6.	Geometry Shader	40
4.6.1.	GS Performance Bottleneck ("maxvertexcount")	42
4.6.2.	A decent use of Geometry Shaders: Point Sprites	42
4.7.	Stream out.....	42
4.7.1.	Skinned Characters Optimization	42
4.7.2.	Blending Morph Targets	43
4.8.	ZCULL and EarlyZ: Coarse and Fine-grained Z and Stencil Culling.	43
Chapter 5. DirectX 10 Considerations.....		45
5.1.	DirectX 10 States and Constants.....	46
	A major cause of poor performance in naïve DirectX 10 ports!	46

5.1.1.	Immutable State blocks	46
5.1.2.	Constant blocks	46
5.1.3.	Don't use global constants!	48
5.1.4.	When to use a tbuffer?	49
5.2.	Resource Management	49
5.2.1.	Resource creation and destruction	49
5.2.2.	Updating resources	50
5.3.	Alpha Test in DirectX 10	51
5.4.	Batching and Instancing	52
Chapter 6. General Advice		53
6.1.	Identifying GPUs	53
6.2.	Hardware Shadow Maps	54
6.3.	Depth Bounds Test (DBT)	55
6.3.1.	Important Notes	55
6.3.2.	API Usage	55
6.3.3.	What is DBT good for?	56
6.4.	FOURCC Codes.....	57
6.4.1.	NULL Rendertarget ("NULL")	57
6.4.2.	Direct DepthBuffer Access ("INTZ" and "RAWZ")	58
Chapter 7. Performance Tools Overview		60
7.1.	PerfKit	60
7.1.1.	PerfHUD	61
7.1.2.	PerfSDK	61
7.1.3.	GLExpert	62
7.1.4.	ShaderPerf	63
7.2.	Shader Debugger	63
7.3.	FX Composer.....	64
Developer Tools Questions and Feedback		65

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.