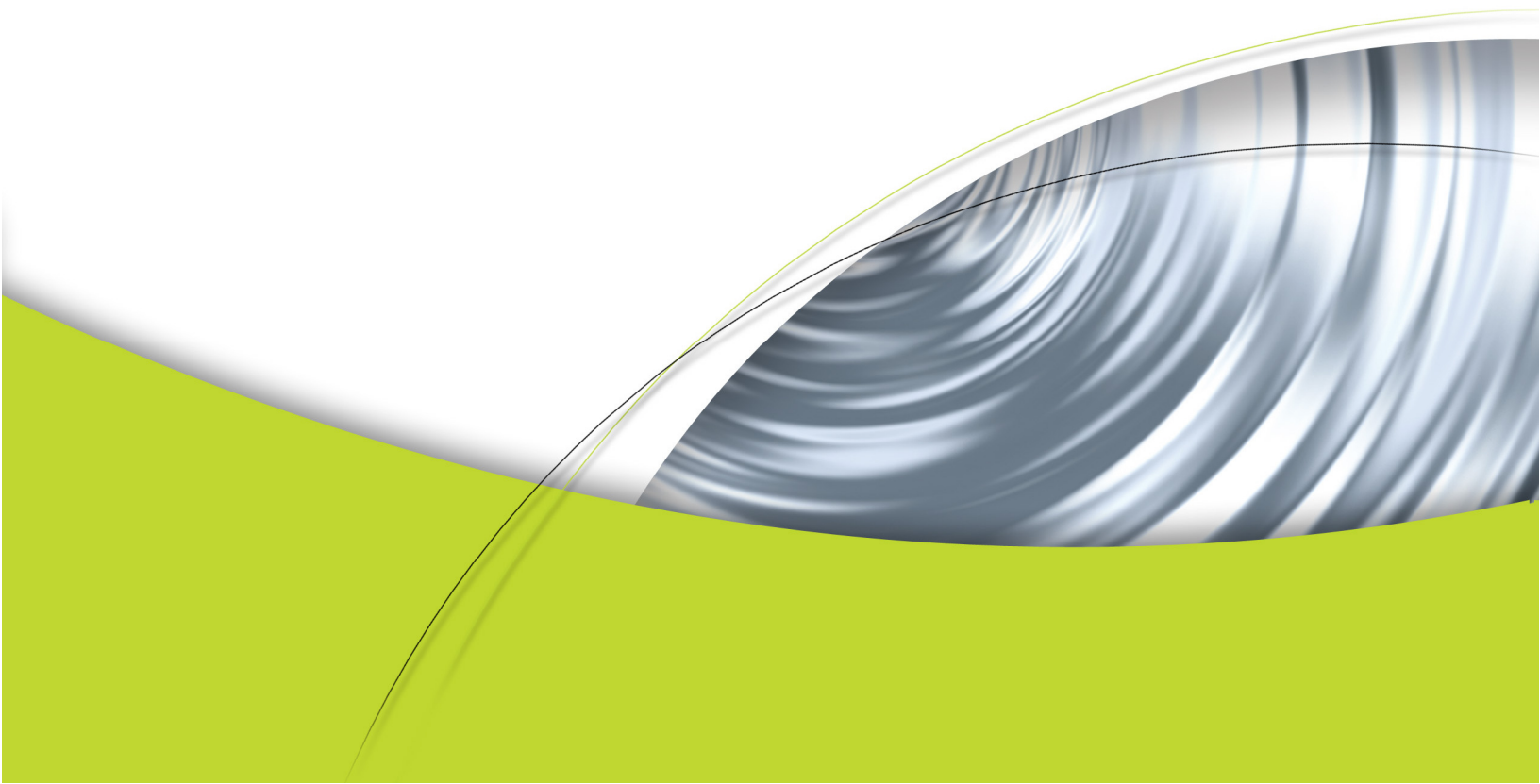




# Cg Toolkit

Cg 3.0  
February 2011  
Release Notes



# Cg Toolkit Release Notes

The Cg Toolkit allows developers to write and run Cg programs using a wide variety of hardware and OS platforms and graphics APIs. Originally released in December 2002, the Toolkit now supports over 30 different DirectX and OpenGL profile targets. It provides a compiler for the Cg language, runtime libraries to use with the OpenGL and DirectX graphics APIs, support for CgFX effect files, example applications, and extensive documentation.

The February 2011 version of Cg 3.0 contains the following improvements:

- ❑ Fixed some bugs in DX11 tessellation profiles
- ❑ Fixed a crash when effects in a context have nearly identical user defined types
- ❑ Fixed a crash when getting default values from unreferenced uniform parameters
- ❑ Fixed problem setting matrices beyond 96 float constants in the vp30 profile
- ❑ Application supplied options now override any set by cgGLSetOptimalOptions
- ❑ Fixed a problem with 'const' variables in the GLSL profiles
- ❑ Added reference manual pages for sampler states

The November 2010 version of Cg 3.0 contains the following improvements:

- ❑ Vertex shaders aren't disabled by tessellation shader state assignments
- ❑ DirectX feature levels 9 and 10 now work with cgD3D11
- ❑ cgCombinePrograms respects the auto compile flag
- ❑ The hlsl10g profile now supports structures in AttribArray
- ❑ cgCopyProgram works once again!
- ❑ Fixed a bug when compound-assignment operators (+=, \*=, etc.) were used on column major matrices
- ❑ tex2Dsize will now work for shadow samplers
- ❑ Resolved a problem with geometry shaders on OSX
- ❑ Corrected our OSX Framework executable path
- ❑ Fixed crash when connecting API created parameters to precompiled (CG\_OBJECT) program parameters
- ❑ No longer try to use typedef keyword in generated GLSL code
- ❑ Fixed a problem in the modf routine
- ❑ Various documentation updates

The 3.0 release of Cg contains the following updates:

- ❑ New OpenGL GPU Program5 profiles
- ❑ New DirectX11 Shader Model 5 profiles
- ❑ Support for tessellation programs
- ❑ Support for up to 32 texture units
- ❑ Unbind routines for D3D programs
- ❑ CgFX buffer routines
- ❑ Dependent parameter routines for CgFX shader arrays
- ❑ Support for texBUF and texRBUF to GLSL profiles
- ❑ Support for GLSL1.30 texelFetch() routines
- ❑ Shadow versions of texBLAHproj functions in the hlsl10f profile
- ❑ Program parameter annotations can be used on entry point functions
- ❑ Improved evaluation engine for expressions in CgFX files
- ❑ Support for the centroid semantic modifier in DX10 profiles
- ❑ Debian .deb packages for Debian and Ubuntu users
- ❑ New examples including:
  - ❑ OpenGL/advanced
    - ❑ cgfx\_buffer\_lighting
    - ❑ cgfx\_tessellation
    - ❑ tess\_bezier
    - ❑ tess\_simple
  - ❑ Direct3D10/advanced
    - ❑ custom\_state\_assignments
    - ❑ include\_string
    - ❑ interpolation\_modifier
  - ❑ Direct3D11/basic
    - ❑ 02\_vertex\_and\_fragment\_program
    - ❑ 03\_uniform\_parameter
    - ❑ 04\_varying\_parameter
    - ❑ 05\_texture\_sampling
    - ❑ 06\_vertex\_twisting
    - ❑ 07\_two\_texture\_accesses
    - ❑ cgfx\_buffer
    - ❑ cgfx\_simple
    - ❑ cgfx\_texture
  - ❑ Direct3D11/advanced
    - ❑ cgfx\_buffer\_lighting

- ❑ cgfx\_bumpdemo
- ❑ cgfx\_bumpdemo\_array
- ❑ cgfx\_latest
- ❑ combine\_programs
- ❑ custom\_state\_assignments
- ❑ gs\_shrinky
- ❑ gs\_simple
- ❑ include\_string
- ❑ interpolation\_modifier
- ❑ Performance improvements and bug fixes (detailed below)

Visit the NVIDIA Cg website at [developer.nvidia.com/page/cg\\_main.html](http://developer.nvidia.com/page/cg_main.html) for complete availability and compatibility information.

Bug reports, issues, and feedback can be sent to [cgsupport@nvidia.com](mailto:cgsupport@nvidia.com). Or join the discussion of Cg at <http://developer.nvidia.com/forums/index.php>.

---

## Supported OS/Hardware Platforms

Cg is available for these platforms:

- ❑ Windows (XP, Vista, Win 7) on x86/x86-64
- ❑ Linux on x86/x86-64
- ❑ Mac OS X (Tiger, Leopard, Snow Leopard) on ppc/i386/x86\_64
- ❑ Solaris 10 on x86/x86-64

The Cg Runtime libraries include:

- ❑ The Cg core runtime library for managing parameters and loading programs
- ❑ The CgGL runtime library for OpenGL based applications
- ❑ The CgD3D11 runtime library for DirectX 11 based applications
- ❑ The CgD3D10 runtime library for DirectX 10 based applications
- ❑ The CgD3D9 runtime library for DirectX 9 based applications
- ❑ The CgD3D8 runtime library for DirectX 8 based applications

---

## Supported Profiles

The Cg compiler currently supports the following hardware profiles:

### OpenGL

- ❑ **gp5tcp** NV\_tessellation\_program5 control program
- ❑ **gp5tep** NV\_tessellation\_program5 evaluation program
- ❑ **gp5gp** NV\_geomemtry\_program5
- ❑ **gp5vp** NV\_vertex\_program5
- ❑ **gp5fp** NV\_fragment\_program5
- ❑ **gp4gp** NV\_geomemtry\_program4
- ❑ **gp4vp** NV\_vertex\_program4
- ❑ **gp4fp** NV\_fragment\_program4
- ❑ **glslg** OpenGL Shading Language (GLSL) for OpenGL 2.0 geometry shader
- ❑ **glslv** OpenGL Shading Language (GLSL) for OpenGL 2.0 vertex shader
- ❑ **glslf** OpenGL Shading Language (GLSL) for OpenGL 2.0 fragment shader
- ❑ **arbvp1** ARB\_vertex\_program 1.0
- ❑ **arbf1** ARB\_fragment\_program 1.0
- ❑ **vp40** ARB\_vertex\_program + NV\_vertex\_program2 option
- ❑ **fp40** ARB\_fragment\_program + NV\_fragment\_program2 option
- ❑ **vp30** NV\_vertex\_program 2.0
- ❑ **fp30** NV\_fragment\_program 1.0

- ❑ **vp20** NV\_vertex\_program 1.0
- ❑ **fp20** NV\_register\_combiners and NV\_texture\_shader

### DirectX 11.0

- ❑ **ds\_5\_0** HLSL11 Domain Shader
- ❑ **hs\_5\_0** HLSL11 Hull Shader
- ❑ **gs\_5\_0** HLSL11 Geometry Shader
- ❑ **vs\_5\_0** HLSL11 Vertex Shader
- ❑ **ps\_5\_0** HLSL11 Fragment Shader

### DirectX 10.0

- ❑ **gs\_4\_0** HLSL10 Geometry Shader
- ❑ **vs\_4\_0** HLSL10 Vertex Shader
- ❑ **ps\_4\_0** HLSL10 Fragment Shader

### DirectX 9.0c

- ❑ **hlslv** HLSL9 Vertex Shader
- ❑ **hlslf** HLSL9 Fragment Shader
- ❑ **vs\_3\_0** Vertex Shader 3.0
- ❑ **ps\_3\_0** Pixel Shader 3.0

### DirectX 9

- ❑ **vs\_2\_x** Extended Vertex Shader 2.0
- ❑ **ps\_2\_x** Extended Pixel Shader 2.0
- ❑ **vs\_2\_0** Vertex Shader 2.0
- ❑ **ps\_2\_0** Pixel Shader 2.0

### DirectX 8 & 9

- ❑ **vs\_1\_1** Vertex Shader 1.1
- ❑ **ps\_1\_3** Pixel Shader 1.3
- ❑ **ps\_1\_2** Pixel Shader 1.2
- ❑ **ps\_1\_1** Pixel Shader 1.1

## Improvements & Bug Fixes

### Improvements

- ❑ `cgGetParameterBufferIndex` and `cgGetParameterBufferOffset` now recompile a program if necessary before returning the queried value. This avoids returning erroneous values when a program is in an uncompiled state.
- ❑ `cgGetFirstProgramAnnotation` will work for non-CgFX programs
- ❑ `cgGetIntAnnotationValues` returns data from unsigned int annotations
- ❑ `cgIsParameterUsed` will now return true for parameters that are assigned to fixed function pipeline texture units.
- ❑ `cgGetParameterBufferOffset` now correctly returns -1 for all parameters that are not in buffers.
- ❑ `cgGetIntAnnotationValues` returns data from unsigned int annotations.
- ❑ `cgGetFirstProgramAnnotation` will work for non-CgFX programs.
- ❑ User-defined constants with the same name as an overloaded standard library function are now handled correctly.
- ❑ The `GenerateMipmap` state assignment is silently ignored for `samplerRECT` rather than causing a GL error.
- ❑ Eliminated spurious warnings from the GLSL profiles about uninitialized variables.
- ❑ UTF-8 byte-order-marker (BOM) are now allowed at the beginning of effect files.
- ❑ Trying to access an undefined struct member now results in a clearer error message.

### Improvements: Documentation

- ❑ **Note:** The Cg Users Manual has **not** been updated for this release.
- ❑ Updated reference manual for the new profiles and entry points.
- ❑ Additional reference manual entries for the default CgFX states.

### Bug Fixes

- ❑ Fixed crash when invalid options strings are passed to the compiler.
- ❑ Fixed crash in `cgGetParameterBufferOffset` and `cgGetParameterBufferIndex`.
- ❑ Fixed issues with C regs and texture lookups in D3D10 translation profile.
- ❑ Fixed `ARB_vertex_array_bgra` support in OpenGL profiles.
- ❑ Fixed a crash when `cgGetEffectParameterBySemantic` is called on an undefined interface.
- ❑ Fixed handling of ENV uniform semantic in GLSL profiles.

---

## Compatibility Notes

Although the 3.0 release of Cg is generally compatible with previous releases, several improvements and other changes may affect existing applications. This section details these potential compatibility issues.

Cg now has explicit enum values for shadow samplers (`CG_SAMPLER1DSHADOW`, `CG_SAMPLER2DSHADOW`, and `CG_SAMPLERRECTSHADOW`) and these will be returned by `cgGetType` for “shadow1D,” “shadow2D,” or “shadowRect.” Previously `cgGetType` would return `CG_SAMPLER1D` or `CG_SAMPLER2D` for these types.

`cgGetProfileString` previously returned `NULL` for `CG_PROFILE_UNKNOWN` but beginning with Cg 3.0 it returns “unknown”.

Global ‘const’ variables in an effect are now treated as true constants and can therefore be used as index values. This should be useful for organizing effects and is expected to provide improved shader optimization. But there’s a potential performance issue if the runtime API is used to change the parameter’s value since the program will then have to be recompiled. If this is a problem you can revert to the old behavior by explicitly supplying the ‘uniform’ keyword in the variable declaration.

There aren’t any other known compatibility issues with programs written against Cg 2.2. For programs written against Cg 2.1 or earlier, refer to the Compatibility Notes section of the release notes for Cg 2.2.



## Known issues

### Known runtime issues

- ❑ `cgGetParameterValues` incurs a penalty in both performance and memory footprint and using it is strongly discouraged. Use `cgGetParameterValue` or `cgGetParameterDefaultValue` instead.
- ❑ `cgCopyEffect` does not work.
- ❑ Loading precompiled code via `CG_OBJECT` in `cgCreateProgramFromFile` doesn't work for shaders which use semantic type modifiers.
- ❑ The Direct3D 8 runtime does not support Cg interfaces.
- ❑ The Cg runtime does not support creating shared parameters containing varying members.
- ❑ Unsized arrays and interface parameters cannot currently be used on the right-hand side of state assignments. Doing so will trigger an error.
- ❑ Values set by `cgGLSetOptimalOptions` will be un-set when the last `CGcontext` is destroyed with `cgDestroyContext`. To work around this, call `cgGLSetOptimalOptions` again after creating more contexts.

### Known compiler issues

- ❑ Long shader programs that make heavy use of interfaces may still result in very long compilation time.
- ❑ Very little error checking is performed on the OpenGL state semantics string (`state.*`); it is just copied to the output assembly. As a result, a typo in the string may compile correctly, and no error will be apparent until the application attempts to load the assembly shader.
- ❑ Error reporting: Some error and warning messages are not as clear as they could be. Some of the issues to be aware of are:
  - ❑ Reported line numbers do not match source code lines when standard library functions are being used
  - ❑ In some cases, errors are not reported in the order they appear in the program
  - ❑ Errors are not reported when constants are out of range for untyped constants.
- ❑ Side-effects in conditional expressions (`?:`) and logical expressions (`&&` and `||`) are always evaluated, regardless of the condition, as specified in the Cg language specification. Hence developers need to watch out for this case.
- ❑ At most one binding semantic per uniform variable is supported by the compiler. Multiple profile-specific binding semantics per uniform variable are not supported.

- ❑ Only loops with a single induction variable are unrolled. Loops that require more than 1 induction variable will fail to compile on older profiles that do not support loops.
- ❑ Local variable arrays which are written to in one block of code, and then read via a non-constant index in a different block will fail to compile on older hardware that does not support this feature. Current hardware supports this feature.
- ❑ Invalid Cg programs can, at times, generate invalid code, instead of a compiler error.

## Known profile-specific issues

- ❑ The Direct3D 11 tessellation profiles don't work.
- ❑ The ps2\* profiles do not support MRTs.
- ❑ Because the underlying hardware support for the fp20 and ps\_1\_\* profiles is quite limited and inflexible, it isn't always possible to compile even seemingly simple Cg programs under these profiles. For more details on these limitations, please see the NV\_register\_combiners and NV\_texture\_shader OpenGL extension specifications, or the DirectX PixelShader 1.\* specifications.
- ❑ The FOG varying input semantic is not supported under the fp20 profile.

## New API

Below is the complete list of the new routines in Cg 3.0. See the Cg Reference Manual for details about each new routine.

**cgCombinePrograms4**  
**cgCombinePrograms5**  
**cgGetBehavior**  
**cgGetBehaviorString**  
**cgGetContextBehavior**  
**cgGetDependentProgramArrayStateAssignmentParameter**  
**cgGetEffectParameterBuffer**  
**cgGetNumDependentProgramArrayStateAssignmentParameters**  
**cgSetContextBehavior**  
**cgSetEffectParameterBuffer**

**cgD3D9UnbindProgram**

**cgD3D10UnbindProgram**

<b>cgD3D11BindProgram</b>	<b>cgD3D11IsProgramLoaded</b>
<b>cgD3D11GetBufferByIndex</b>	<b>cgD3D11LoadProgram</b>
<b>cgD3D11GetCompiledProgram</b>	<b>cgD3D11RegisterStates</b>
<b>cgD3D11GetDevice</b>	<b>cgD3D11SetDevice</b>
<b>cgD3D11GetIASignatureByPass</b>	<b>cgD3D11SetManageTextureParameters</b>
<b>cgD3D11GetLastError</b>	<b>cgD3D11SetSamplerStateParameter</b>
<b>cgD3D11GetLatestDomainProfile</b>	<b>cgD3D11SetTextureParameter</b>
<b>cgD3D11GetLatestGeometryProfile</b>	<b>cgD3D11SetTextureSamplerStateParameter</b>
<b>cgD3D11GetLatestHullProfile</b>	<b>cgD3D11TranslateCError</b>
<b>cgD3D11GetLatestPixelProfile</b>	<b>cgD3D11TranslateHRESULT</b>
<b>cgD3D11GetLatestVertexProfile</b>	<b>cgD3D11TypeToSize</b>
<b>cgD3D11GetManageTextureParameters</b>	<b>cgD3D11UnbindProgram</b>
<b>cgD3D11GetOptimalOptions</b>	<b>cgD3D11UnloadProgram</b>
<b>cgD3D11GetProgramErrors</b>	
<b>cgD3D11IsProfileSupported</b>	

---

## Release Types

Cg is released in two forms:

1. The Cg Toolkit provides a complete Cg Software Development Kit (SDK) including documentation, examples, standalone compiler, headers and libraries.
2. Cg Binary Distributions provide updated redistributable libraries that Cg-based applications can ship with.

SDK versions are released as platform specific installers containing the full toolkit (libraries, documentation, examples, etc.) They can be downloaded from

[http://developer.nvidia.com/object/cg\\_toolkit.html](http://developer.nvidia.com/object/cg_toolkit.html)

Binary distributions contain only the libraries, and all supported platforms are bundled in a single file. The libraries supplied in a binary distribution should be feature-for-feature and bug-for-bug compatible across all the platforms supported by a given distribution (meaning are all compiled from the same source code). Cross-platform software vendors are encouraged to redistribute Cg libraries from a single binary distribution to minimize platform variances in Cg.

Cg binary distributions can be found at

<http://developer.nvidia.com/object/cg-redistributable-binaries.html>

## Distribution License

The docs directory contains a file license.pdf providing a non-exclusive, world-wide, royalty free license for redistributing Cg with your applications. See this license for details.

## Release History

The following table summarizes release dates and library versions for Cg releases:

Cg Release Name	Release Date	Library Version
SDK2	11/15/10	3.0.0015
SDK1	07/22/10	3.0.0007
beta	04/22/10	3.0.0005

The Cg library version is returned by `cgGetString(CG_VERSION)`

## Change History

### 3.0.0015

- ❑ Vertex shaders aren't disabled by tessellation shader state assignments
- ❑ DirectX feature levels 9 and 10 now work with cgD3D11
- ❑ `cgCombinePrograms` respects the auto compile flag
- ❑ The hlsl10g profile now supports structures in `AttribArray`
- ❑ `cgCopyProgram` works once again!
- ❑ Fixed a bug when compound-assignment operators (`+=`, `*=`, etc.) were used on column major matrices
- ❑ `tex2Dsize` will now work for shadow samplers
- ❑ Resolved a problem with geometry shaders on OSX
- ❑ Corrected our OSX Framework executable path
- ❑ Fixed crash when connecting API created parameters to precompiled (`CG_OBJECT`) program parameters
- ❑ No longer try to use typedef keyword in generated GLSL code

### 3.0.0007

- ❑ Debian .deb packages for Debian and Ubuntu users
- ❑ Fixed handling of ENV uniform semantic in GLSL profiles
- ❑ Support centroid semantic modifier in DX10 profile
- ❑ Added support for `texBUF` and `texRBUF` to GLSL profiles
- ❑ Added support for GLSL1.30 `texelFetch()` routines
- ❑ `cgGetIntAnnotationValues` now returns data from unsigned int annotations
- ❑ `cgIsParameterUsed` now returns true for parameters that are assigned to fixed function pipeline texture units

- ❑ Add support for `cgGetFirstProgramAnnotation` for non-CgFX programs
- ❑ Add support for program parameter annotations for entry point function
- ❑ Silently ignore `GenerateMipmap` state assignment for `samplerRECT`

### 3.0.0005

- ❑ OpenGL GPU Program5 profiles
- ❑ DirectX11 Shader Model 5 profiles
- ❑ Support for tessellation programs
- ❑ Support for up to 32 texture units
- ❑ Unbind routines for D3D programs
- ❑ CgFX buffer routines
- ❑ Dependent parameter routines for CgFX shader arrays
- ❑ Shadow versions of `texBLAHproj` functions in the `hlsl10f` profile
- ❑ handle user-defined constants with same name as an overloaded `stdlib` function

## **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 and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation.

Microsoft, Windows, the Windows logo, and DirectX are registered trademarks of Microsoft Corporation.

OpenGL is a trademark of SGI.

Other company and product names may be trademarks of the respective companies with which they are associated.

## **Copyright**

Copyright © 2004-2010 NVIDIA Corporation. All rights reserved.



**NVIDIA.**

NVIDIA Corporation  
2701 San Tomas Expressway  
Santa Clara, CA 95050  
[www.nvidia.com](http://www.nvidia.com)