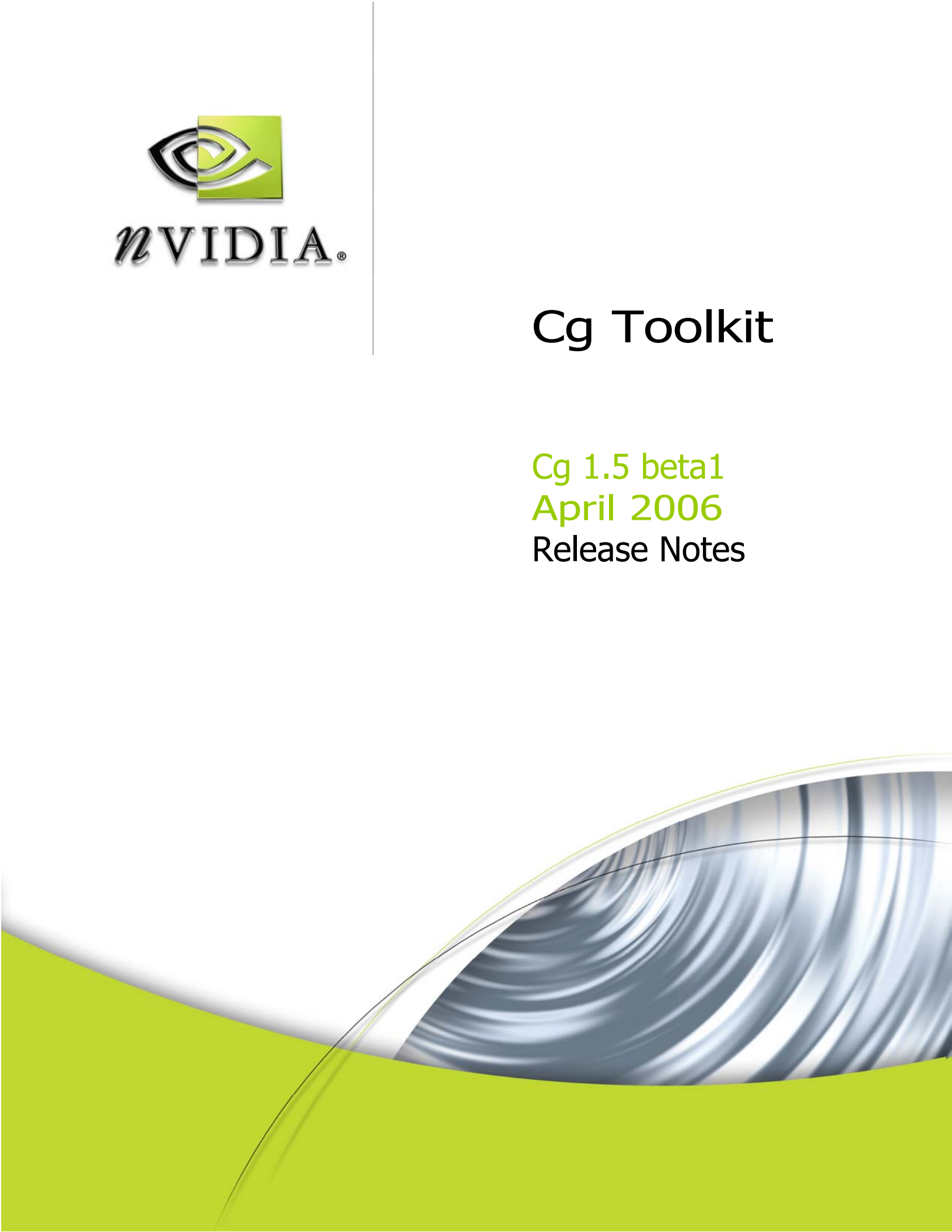


# Cg Toolkit

## Cg 1.5 beta1
## April 2006
## Release Notes

# Cg Toolkit Release Notes

The Cg Toolkit allows developers to write and run Cg programs using a wide variety of hardware platforms and graphics APIs.

Originally released in December 2002, the Toolkit now supports over 20 different DirectX and OpenGL profile targets. It provides a compiler for the Cg language, runtime libraries for use with the OpenGL and DirectX graphics APIs, support for the CgFX effect files, example applications, and extensive documentation.

This version 1.5 beta 1 release of Cg incorporates new functionality:

- New OpenGL profiles for the OpenGL Shading Language (GLSL). `glslv` is the new vertex profile; `glslf` is the new fragment profile.
- New API for programmatic effect generation.
- New API for combining programs from multiple domains.
- Improved examples for both OpenGL and Direct3D.
- Direct3D profiles for Shader Model 3.0 (`vs_3_0` and `ps_3_0`) are now supported.
- The CgFX API works for Direct3D now.

Functionality from Cg 1.4.1 is still present:

- Significantly improved code generation in most of the profiles
- Complete rewrite of CgFX, including an API integrated with the Cg runtime API, support for extensible, cross-platform graphics state management, and support for unsized arrays and interfaces.
- Substantial improvements in compiling many long shaders, especially those that make heavy use of interfaces.
- The Cg runtime is now substantially faster in many cases
- Cg 1.4.1 now ships with native implementations for Win32, Win64, Linux (32-bit and 64-bit), and MacOS 10.3 (Panther) and 10.4 (Tiger)

With Cg 1.5, the Direct3D-specific Cg runtime libraries support shared parameters as well as a native D3D state manager for CgFX. This release also provides Direct3D profiles for Shader Model 3.0.

With minor exceptions, Cg 1.4, Cg 1.3 and Cg 1.2 applications should work with Cg 1.5 without the need to recompile the program. See "Compatibility Notes," below, for more information.

Applications and effects that used previous versions of CgFX will have to be modified to use the new CgFX API. If the supplied OpenGL state manager for CgFX is used, the

effect files themselves must be modified. The effects must be changed to use OpenGL state assignments, rather than the D3D state assignments supported in prior versions of CgFX.

Cg is available for a wide variety of hardware and OS platforms (as mentioned above). Please visit the NVIDIA Cg website at developer.nvidia.com/Cg for complete availability and compatibility information.

Please report any bugs, issues, and feedback to NVIDIA by email at cgsupport@nvidia.com. We will expeditiously address any reported problems.

# Compatibility Notes

Although the 1.5 beta1 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.

In the 1.2 and 1.3 releases of Cg, the Cg compiler would always generate generic attribute names (e.g., "ATTR0") for varying input parameters, even if conventional semantic names (e.g., "POSITION") were specified under OpenGL profiles. Since release 1.4.1, the type of attribute now matches the type of semantic. As a result, applications that explicitly assumed and checked for generic attribute names may need to be modified to handle both generic and conventional attributes.

In this release, the Cg compiler uses an improved, more efficient constant register allocation scheme. As a result, uniform parameters in compiled programs may no longer reside in the same constant registers as they had in previous releases of Cg. Applications that (incorrectly) assume a particular constant register numbering or order should be modified to either remove this assumption, or to use register allocation semantics (e.g., "register(C0)") to implement the layout assumed by the application.

CgFX has been redesigned and re-implemented. As noted above, existing applications that use CgFX must be modified if the new API is to be used. In addition, if you wish to use the supplied "native" OpenGL state manager, which assumes that OpenGL-style state names and values are used in effects, existing effects must be modified to use OpenGL-style states. See the "Introduction to CgFX" chapter in the Cg Users Manual for more details. If you have questions or problems related to porting existing CgFX applications, please contact NVIDIA developer relations.

# Supported Profiles and Platforms

The Cg compiler currently supports the following hardware profiles:

OpenGL

- glslv (OpenGL 2.0 vertex shader)
- glslf (OpenGL 2.0 fragment shader)
- arbvp1 (ARB_vertex_program 1.0)
- arbfp1 (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 8 & 9

- vs_1_1 (Vertex Shader 1.1)
- ps_1_1, ps_1_2 and ps_1_3 (Pixel Shader 1.1, 1.2, 1.3)

### DirectX 9

- vs_2_0 and vs_2_x (Vertex Shader 2.0 and Extended VS 2.0)
- ps_2_0 and ps_2_x (Pixel Shader PS 2.0 and Extended PS 2.0)

### DirectX 9.0c

- vs_3_0 (Vertex Shader Model 3.0)
- ps_3_0 (Pixel Shader Model 3.0)

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 CgD3D8 runtime library for DirectX 8 based applications
- The CgD3D9 runtime library for DirectX 9 based applications
- Since Cg 1.4, CgFX is incorporated directly into the Cg Runtime libraries. With Cg 1.5, CgFX supports Direct3D as well as OpenGL.

# Improvements & Bug Fixes

## Improvements

- CgFX supports D3D-style state names
- Many changes to the infrastructure in both the compiler and runtime have been made in order to improve both performance and reliability

## Improvement: CgFX rewrite

CgFX has been completely rewritten. Major goals of this rewrite:

- Improve efficiency and reliability of the CgFX Runtime
- Provide a means of supporting extensible, cross-graphics-API states and state managers
- Create a better mapping between CgFX files and OpenGL state, for those applications that focus exclusively on OpenGL
- Allow for compile-time execution of Cg expressions through a Cg virtual machine
- Portability across more platforms

CgFX is now more stable, faster, and feature-rich.

**NOTE**: This version of CgFX is not API-compatible, nor effect-file-compatible, with previous releases. NVIDIA is committed to supporting developers who have been using previous versions of CgFX. If you need help porting your application or effects to the new version of CgFX, please contact your NVIDIA Developer Support contacts.

## Improvement: Cg Runtime efficiency

The Cg Runtime has been tuned to remove a significant amount of overhead from the runtime calls. In many cases, CPU overhead in the Cg runtime has been reduced on the order of 50%.

## Improvement: Cg compile times

Many Cg programs should now compile faster than in previous releases. This is especially true of long shaders, and shaders that make heavy use of Cg's "interfaces" feature. Further improvements are still being worked on for a future Cg release.

## Improvement: Windows Installer

Cg 1.5 for Windows now installs a more complete set of example code.

The Cg 1.4.1 installer for Windows is thoroughly updated and now based on Inno Setup (rather than InstallShield). There is no longer a separate Cg 64-bit installer for Windows. The single Cg installer can optionally install 64-bit libraries for x86-64 systems. The new installer will automatically first remove a prior InstallShield-based installer.

The Cg 1.4.1 installer automatically makes the necessary updates to your `Path`, `CG_BIN_PATH`, `CG_INCLUDE_PATH`, and `CG_LIB_PATH` environment variables.

The Cg 1.4.1 installer has more control over the selected components for installation. For example, you can skip installation of documentation, examples, or 64-bit support. Everything is installed by default except the 64-bit libraries and executables.

A selectable install component (installed by default) provides Visual Studio integration for Cg for Visual C 6.0 and Visual Studio .NET 7.1.

## Improvement: Documentation

- Note: The Cg Users Manual has not yet been updated for Cg 1.5.
- The Cg 1.4.1 Users Manual in PDF format is included in the install at `docs/CgUsersManual.pdf` or navigate the Windows Start menu to find *Start->All Programs->NVIDIA Corporation->Cg Toolkit->User's Manual*.
- Reference manual pages for the Cg runtime API, profiles, and Cg standard library are now included in HTML and PDF form. Find the HTML pages in the docs/html directory. Find the `CgReferenceManual.pdf` in the docs directory or navigate the Windows Start menu to find *Start->All Programs->NVIDIA Corporation->Cg Toolkit->Reference Manual* (select the PDF icon).
- The Microsoft HTML Help file is updated to include profile and standard library documentation. Navigate the Windows Start menu to find *Start->All Programs->NVIDIA Corporation->Cg Toolkit->Reference Manual* (select the HTML Help icon).

## Removed features

- We have dropped support for the previous CgFX API
- We no longer provide debug `cgD3D8d.dll` and `cgD3D9d.dll` libraries.

## Bug Fixes

- In CgFX, too many bugs have been fixed to list them here.
- All known memory leaks in CgFX and the Cg Runtime have been fixed
- The determinant standard library routine for a 4x4 matrix is correct now.

- Constant loading issues with the fp20 profile are resolved.

# Known issues

## Known runtime issues

- The DirectX 8 runtime has not been updated to support the Cg 1.2 'interfaces' feature.
- The Cg runtime does not currently support created 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(...) can be un-set after a call to cgDestroyContext( ). To work around this issue, cgGLSetOptimalOptions( ) should be called after each call to cgDestroyContext( ), if more Cg contexts are going to be subsequently created.
- Error reporting should be improved in many cases.
- More OpenGL state still needs to be exposed through CgFX state assignments. If you have specific feedback about missing state, please speak to your Developer Support contact.

## Known compiler issues

- The 1.4.1 release of Cg is more accurate at detecting when program parameters are not used.
- Long shader programs that make heavy use of interfaces may still see very long compiler times.
- 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, and currently warnings are not always issued. Hence developers need to watch out for this case.

- Only one return statement is allowed per function. There is an error issued if there is any unreachable code. Return statements in if/for blocks are not supported.
- 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 single induction variables are unrolled. Loops that require more than 1 induction variable will fail to compile.
- 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 cause the compiler to crash. Current hardware does not support this feature, but the compiler should not crash.
- Invalid Cg programs can, at times, generate invalid code, instead of a compiler error.

## Known profile-specific issues

- The ps2* profiles do not yet support MRTs
- Because the underlying hardware support for the fp20 and ps_1_* profiles is quite limited and inflexible, it is not always possible to compile seemingly simple Cg programs under these profiles. For more details on these limitations, please see the NV_register_combiners, NV_texture_shader OpenGL extension specifications, or the DirectX PixelShader 1.* specifications.
- The FOG varying input semantic is not yet supported under the fp20 profile. (also true in Cg 1.2 and 1.3)

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