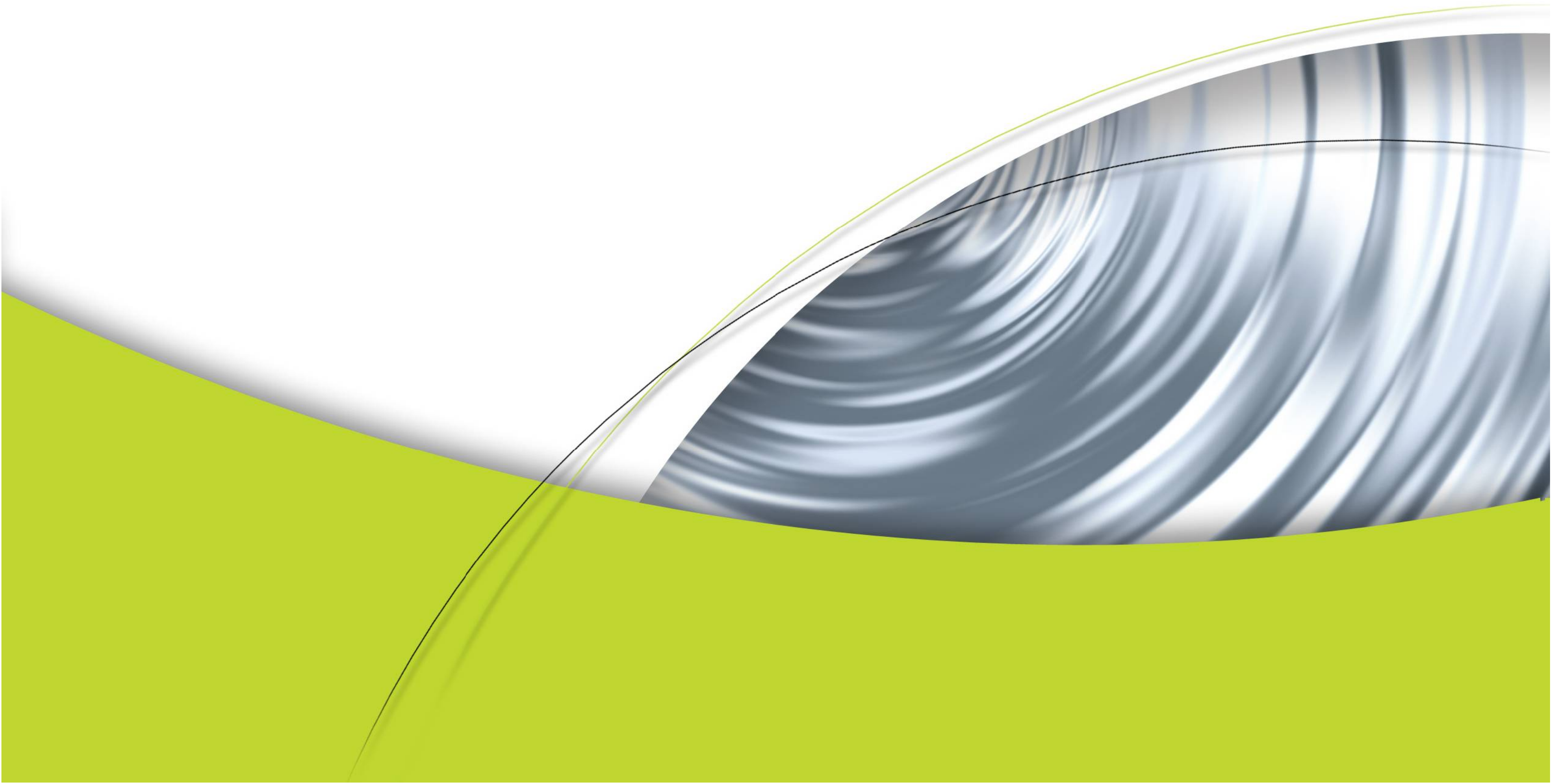
Cg Toolkit

Cg 1.4 rc 1

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, runtime libraries for CgFX, example applications, and extensive documentation.

This release candidate ('rc') build of the NVIDIA Cg 1.4 Toolkit is a development snapshot intended to give early adopters a preview of the new features in Cg 1.4, and to solicit developer feedback.

This 1.4 rc build of Cg introduces several significant changes and features:

- Significantly improved code generation in most of the profiles
- Complete rewrite of CgFX. CgFX now works on all platforms, and works on non-NVIDIA OpenGL drivers. Substantial documentation for CgFX will be coming in a subsequent build of Cg 1.4
- CgFX now supports unsized arrays and Interfaces
- Substantially faster at compiling many long shaders, especially those that make heavy use of Cg's "Interfaces" feature
- The Cg Runtime is now substantially faster in many cases
- Cg 1.4 now ships with native implementations for Win32, Win64, Linux (32-bit and 64-bit), and MacOS 10.3 (Panther) and 10.4 (Tiger)

Note that some of this new functionality is currently only supported when using the CgGL OpenGL runtime library. The Direct3D-specific Cg runtime libraries currently do not support shared parameters, for example. We have also not yet provided Direct3D profiles for Shader Model 3.0.

All Cg 1.2 or Cg 1.3 programs should work with Cg 1.4 without the need to recompile the program. The compiler, runtime, and standard library changes were all made to be backward-compatible. However, programs which used previous versions of CgFX will require changes, both to the calling application code, and also to the effect files

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.

Supported Profiles and Platforms

The Cg compiler currently supports the following hardware profiles:

OpenGL

- arbvp1 (ARB_vertex_program 1.0)
- arbfp1 (ARB_fragment_program 1.0)
- vp40 (ARB_vertex_program + NV_vertex_program2 option)
- vp30 (NV_vertex_program 2.0)
- fp40 (ARB_fragment_program + NV_fragment_program2 option)
- 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/ps_2_a (Pixel Shader PS 2.0 and Extended PS 2.0)

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
- CgFX has been incorporated directly into the Cg Runtime libraries

Improvements & Bug Fixes

Improvements

- The complete rewrite of CgFX is the most substantial change to this release
- Many changes to the infrastructure in both the compiler and runtime were also made, though, in an effort to improve both performance and reliability

Improvement: CgFX rewrite

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

- Create a more reliable base of software
- Improve efficiency of the CgFX Runtime
- Create a better mapping between CgFX files and OpenGL state
- Allow for compile-time execution of Cg expressions through a Cg virtual machine
- Portability across more platforms

CgFX is now very stable, fast, and featureful. Preliminary and interim documentation on the new CgFX APIs, and state vectors, is provided with this release package. We are working on improving this documentation for the official Cg 1.4 release.

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 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, programs can run as much as 20% to 30% faster, if those programs previously spent many CPU cycles in the Cg Runtime calls

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.

Removed features

- We have dropped support for the previous CgFX API and effect file formats

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

Known issues

Known runtime issues

- The DirectX 8 and DirectX 9 runtimes have not yet been updated to support the Cg 1.2 'Interfaces' feature.
- CgFX support is not yet complete in the DirectX 8 and DirectX 9 runtimes
- cgCreateEffectFromFile() will return NULL and set the UNSUPPORTED_PROFILE error if the effect file being loaded contains some techniques that are not supported by the currently-active graphics device. We will attempt to fix this before the 1.4.0 release.
- Cg will generate invalid assembly in some cases, for programs that use varying and mixed structs with Interfaces. We will attempt to fix this before the 1.4.0 release.
- Unsized arrays and interface parameters cannot currently be used on the right-hand side of state assignments. Doing so will trigger an error.
- The 'cg_explicit' and 'cgGL_explicit' libraries are not currently supported.
- 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.
- Connecting an array of size 0 to a resizeable array will cause internal failures within the Cg Runtime.
- More work still needs to be done on error reporting.
- 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

- Long shader programs that make heavy use of Interfaces may still see very long compiler times. We are working on addressing this issue, but will likely address this after 1.4.0 ships.

- 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.
- All matrices are assumed to be row-major. Currently, column-major matrices 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)

Known documentation issues

- The Cg documentation is being overhauled for the 1.4 release. Current docs are largely outdated.

Known installer issues

- The Cg 1.4 installer currently only ‘upgrades’ a previous install of Cg 1.x. As a result, the installer will remove or overwrite previous versions of the Cg Toolkit when installing this version.

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