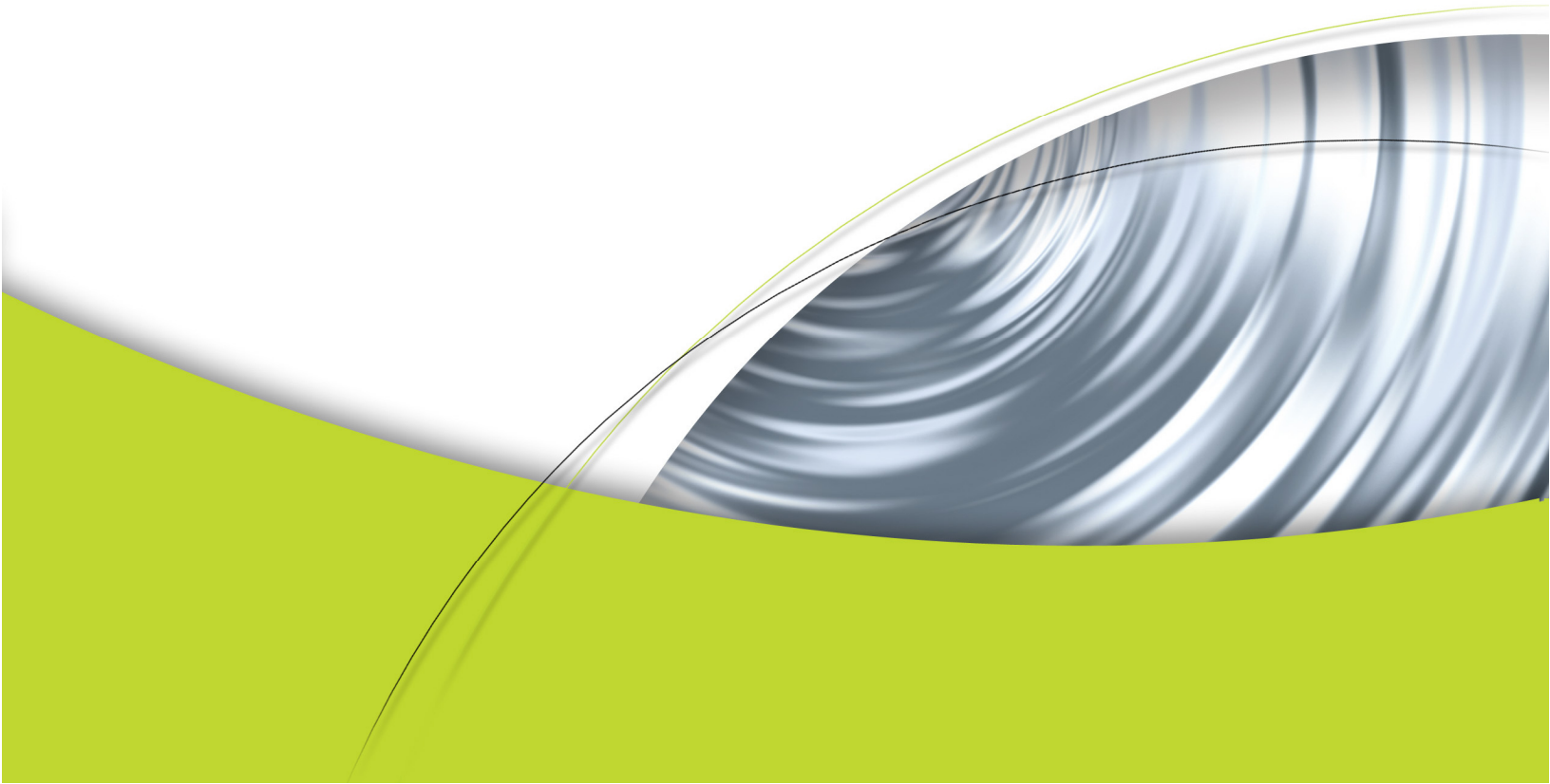
Cg Toolkit

Cg 2.2
February 2010
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 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.

The January 2010 version of Cg 2.2 has these changes:

- Require EXT_gpu_shader4 in GLSL when using bit shift/mask instructions
- Modified example gs_simple to explicitly use the GLSL profiles if supported
- HLSL semantic VFACE is now accepted as an alias for semantic FACE
- Improved our handling of extensions on older versions of OpenGL
- Various performance improvements
- Enhanced cgfxcat to work for program files as well as effect files
- Fixed some compiler crashes with malformed shaders
- Fixed a crash in cgGetParameterBufferIndex and cgGetParameterBufferOffset
- Fixed a bug in cgGetPassProgram for combined programs
- Fixed a problem with geometry shaders on Solaris

The October 2009 version of Cg 2.2 contains the following improvements:

- Allow compiler options in effect compile statements
  - e.g. VertexProgram = compile vp40 "-po PosInv=1" shader();
- Better performance when running on multicore CPUs
- Choosing the "latest" profile is now deferred until effect validation
- Improved the inverse matrix computation in cgGLSetStateMatrixParameter
- Better memory behavior when a program is repeatedly recompiled
- Fixed an issue when using PSIZE semantic with ps_3_0 and ps_4_0 profiles
- cgCombinePrograms now works with CG_OBJECT programs
- cgGetNextProgram was always returning 0
- Fixed a problem with effect parameters and cgGLGetTextureEnum
- Allow comments prior to the shader version in D3D asm blocks of an effect
- HLSL10: Mark globally scoped temporaries 'static' to keep them out of constant buffers

- HLSL10: Allow any semantic for varyings provided the semantics match between stages
- HLSL10: Fix handling of TEXUNITn semantic
- HLSL10: Added support for arrays of samplers
- HLSL10: Empty structs for uniform parameters crashed the D3D compiler
- Fixed a problem when connecting parameters of type string
- Corrected issues in the fp20 profile on Solaris
- Now using MesaGLUT-7.5 for GLUT on Windows

The 2.2 release of Cg incorporates the following updates:

- DirectX10 and GLSL geometry profiles (gs_4_0 AND glslg)
- Support for "latest" profile keyword in CgFX compile statements
- Additional API routines (see **New API** for a complete list)
- Support for pack_matrix() pragma
- Arrays of shaders can now be used in CgFX files
- Libraries for 64 bit Solaris development
- Migrated the OpenGL examples onto GLEW
- New examples including:
  - examples/Direct3D10/advanced
    - combine_programs
    - gs_shrinky
    - gs_simple
  - examples/OpenGL/advanced
    - cgfx_latest
  - examples/Tools
    - cgfxcat
    - cginfo
- New documentation
  - Updated reference manual pages for new profiles and entry points
- Bug fixes

Visit the NVIDIA Cg website at 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.

Supported OS/Hardware Platforms

Cg is available for these platforms:

- Windows 32
- Windows 64
- Linux x86
- Linux x86-64
- MacOS 10.4 (Tiger)
- MacOS 10.5 (Leopard)
- Solaris 10 x86
- Solaris 10 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 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

- **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
- **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 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

- Now support pack_matrix() pragma to specify matrix layout order
- Arrays of shaders can now be used in CgFX files
- Added ability to query the supported profiles at runtime
- Added API to programmatically control profiles returned as "latest"
- Added API to discover the enumerants associated with a state
- Migrated OpenGL examples onto GLEW
- New program for dumping CgFX files (cgfxcat)
- New program for dumping Cg's version string (cginfo)
- Assign GLSL texture resources as soon as a program is compiled rather than waiting until it gets loaded

Improvements: Documentation

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

Bug Fixes

- Fixed the GL profiles to actually unbind programs in cgGLUnbindProgram().
- Fixed a problem with cgGLSetOptimalOptions() on non-NVIDIA GPUs.
- Allow cgSetParameter* functions to work for varying parameters. We don't recommend doing this, but it worked in the past and now it works once again.
- Added an ATI_draw_buffers profile option for the GLSL profiles.
- Generate tex2DGrad() functions for GLSL profile when using tex2D() function with derivatives.
- Fixed the DCL statements for arrays in vs 2.0+ shaders (which were always incorrectly assumed to have a TEXCOORD semantic).
- Fixed a bug in scanning files for #include statements when the last line of the file being included ended with an #endif and was missing an end-of-line character.

Compatibility Notes

There aren't any known compatibility issues with programs written against Cg 2.1. For programs written against Cg 2.0 or earlier, refer to the Compatibility Notes section of the release notes for Cg 2.1.

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.
- **cgCopyProgram** and **cgCopyEffect** do not work.
- Loading precompiled code via **CG_OBJECT** in **cgCreateProgramFromFile** doesn't work for shaders which use semantic type modifiers.
- The DirectX 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 cgDestroyContext() if more contexts are going to be created.

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 GLSL geometry profile has issues on OSX and Solaris.
- 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 yet supported under the fp20 profile.

New API

Cg 2.2 adds new API to query supported profiles, control the latest profile keyword, discover registered state enumerants, as well as other utility functions. Here is the complete list of the new routines:

cgGetDomain

cgGetDomainString

cgGetMatrixParameterOrder

cgGetNumStateEnumerants

cgGetNumSupportedProfiles

cgGetParameterClassEnum

cgGetParameterClassString

cgGetProfileProperty

cgGetProgramDomain

cgGetStateEnumerant

cgGetStateLatestProfile

cgGetSupportedProfile

cgIsProfileSupported

cgSetStateLatestProfile

cgGLGetOptimalOptions

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

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	09/29/09	2.2.0010
SDK1	03/31/09	2.2.0006
beta	02/04/09	2.2.0004
SDK3	02/17/09	2.1.0017

The Cg library version is returned by cgGetString(CG_VERSION)

Change History

2.2.0010

- Allow compiler options in effect compile statements
- Better performance when running on multicore CPUs
- Choosing the "latest" profile is now deferred until effect validation
- Improved the inverse matrix computation in cgGLSetStateMatrixParameter
- Fixed an issue when using PSIZE semantic with ps_3_0 and ps_4_0 profiles
- cgCombinePrograms now works with CG_OBJECT programs
- cgGetNextProgram was always returning 0
- Fixed a problem with effect parameters and cgGLGetTextureEnum
- Allow comments prior to the shader version in D3D asm blocks of an effect
- Corrected assorted problems in the HLSL10 profiles
- Fixed a problem when connecting parameters of type string
- Corrected issues in the fp20 profile on Solaris

2.2.0006

- Support for pack_matrix() pragma
- Arrays of shaders can now be used in CgFX files
- Libraries for 64 bit Solaris development

2.2.0004

- DirectX10 and GLSL geometry profiles (gs_4_0 AND glslg)
- Support for "latest" profile keyword in CgFX compile statements
- Migrated the OpenGL examples onto GLEW

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

ERROR: undefined
OFFENDING COMMAND: ‘~

STACK: