User's Manual

Release Notes

Release 1.1
February 2003

Updates

Any changes, additions, or corrections will be posted at the NVIDIA Cg Web site:

http://developer.nvidia.com/Cg

Refer to this site often to keep up on the latest changes and additions to the Cg language.

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

Cg Toolkit Release Notes

The Cg Toolkit, originally released in December 2002, supports 14 different DirectX 8, DirectX 9, and OpenGL profile targets. The Cg Toolkit allows developers to write Cg programs for a wide variety of hardware platforms and graphics APIs.

Release 1.1 of the Cg Toolkit adds these benefits:

- CgFX beta support for DirectX 8, DirectX 9, and OpenGL
- Fixes for bugs in Release 1.0 (December 2002)
- Additional optimizations

These benefits are described more fully in "Changes for Cg Toolkit Release 1.1" on page 2.

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

Supported Profiles and Runtime Libraries

The Cg compiler currently supports these profiles:

- **`vs_1_1`** for DirectX 8 and DirectX 9, targeting VertexShader VS 1.1
- **`vs_2_0`** and **`vs_2_x`** (known collectively as **`vs_2_*`**) for DirectX 9, targeting VertexShader VS 2.0 and Extended VS 2.0
- **`ps_1_1`**, **`ps_1_2`**, and **`ps_1_3`** (collectively, **`ps_1_*`**) for DirectX 8 and DirectX 9, targeting PixelShader PS 1.1, 1.2, and 1.3
- **`ps_2_0`** and **`ps_2_x`** (collectively, **`ps_2_*`**) for DirectX 9, targeting PixelShader PS 2.0 and Extended PS 2.0
- **`arbvp1`** for OpenGL, targeting ARB_Vertex_Program 1.0
- **`arbfp1`** for OpenGL, targeting ARB_Fragment_Program 1.0
- **`vp20`** and **`vp30`** for OpenGL, targeting NV_Vertex_program 1.0 and NV_Vertex_program 2.0
- **`fp30`** for OpenGL, targeting NV_Fragment_Program 1.0

- **fp20** for OpenGL, targeting NV_register_combiners and NV_Texture_shader

Cg includes these runtime libraries:

- Core runtime library for parameter management and loading programs
- Runtime library for applications based on DirectX 8
- Runtime library for applications based on DirectX 9
- Runtime library for applications based on OpenGL
- CgFX beta runtime libraries for applications based on DirectX 8, DirectX 9, and OpenGL.

In response to developer feedback, the Cg runtime libraries have been designed to have an intuitive API with a clean and consistent interface and to expose new features. To help the transition from the runtime API prior to that of Cg Toolkit 1.0, a runtime transition guide has been included in this package.

Changes for Cg Toolkit Release 1.1

New Feature

The CgFX beta runtime libraries for DirectX 8, DirectX 9, and OpenGL. The library supports **vs_1_1**, **vs_2_0**, **vs_2_x**, **ps_2_0**, **ps_2_x**, **arbvp1**, and **arbfp1** as compile targets.

Deprecated Feature

The mechanism to define connector structures using tags, such as **vertex2fragment** or **fragment2framebuffer**, has been deprecated.

Fixed Bugs

- Fixed compilation of **discard** statement with booleans so it generates correct assembly.
- Fixed **clip()** functionality.
- Fixed **pack*()** and **unpack*()**, which are supported in **fp30** only.
- Input varying parameters may now be declared **const**.
- Global variables are now initialized properly in **vp30**, **arbfp1**, and **vp20**.
- Negation now works properly when used in conjunction with **saturate()**.
- **COLOR** semantic names are now consistent in **arbvp1**.

- Fixed parameter binding issues in **arbvp1** and **arbfp1**.
- Fixed problems with the assignment of arrays of **sampler** to texture units.
- Improved the test for a texture unit being bound to multiple **sampler** instances in **ps_1_***.
- Constant binding conflict issues have been fixed in **ps_1_***.
- Problems using **uniform** variables with the ternary operator have been fixed in **ps_1_***.
- Fixed handling of scalar program output for **vs_1_1**.
- Using **glstate** matrices and lights no longer produces invalid assembly.
- **CG_PROFILE_UNKNOWN** may now be used in **cgCreateProgram()**.
- Fixed **NULL** pointer issue with **cgGLSetParameterPointer()**.
- Variants of **cgGLGetMatrixParameterArray*()** are now available in the cgGL library.
- Internal C++ symbols in runtime libraries are now hidden under Linux.

Known Issues and Unimplemented Features

The issues and features listed below will be addressed in future releases.

Features Not Yet Supported in the Preprocessor

The **#** and **##** macro operators are not supported.

Known Issues in CgFX

CgFX is still in a developmental stage, and there are known issues. As CgFX evolves and more functionality and more support for other platforms is added, these known issues will be addressed. For more information on CgFX, please refer to *CgFX_Overview.pdf* in the docs directory of the Cg Toolkit.

Known Issues in the Language Implementation

- Error reporting
  Some issues to be aware of are
  - Reported line numbers do not match source code lines when Standard Library functions are 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.
- The wording of some error messages could be improved.

Work on these issues is in progress.

You should also take care not to use keywords as identifiers in your programs. (Using **`in`** or **`out`** as a variable name is a common pitfall.) Errors issued in such cases may be difficult to understand.

Unwritten **`out`** parameters contain undefined values, which can lead to bugs in the application. The Cg compiler issues a warning in this case.

Side-effects in conditional expressions (**`?:`**) and logical expressions (**`&&`** and **`||`**) are always evaluated, regardless of the condition. Currently, warnings are not always issued and you need to watch out for these cases.

It is legal to overload Standard Library functions; the compiler does not issue any warnings in this case. Hence, you need to watch out for conflicts in names of user-defined and Standard Library functions.

- Unless defined as **`static`**, **`const`** globals are not constant folded, which may lead to inefficient code. However, static constant globals and locals are constant folded. One way to avoid inefficient code is to use static constant variables whenever possible.
- There is no support for **`inout`** parameters to the entry function of a program, but **`inout`** parameters to non-entry functions work well.
- Semantics is not supported on varying array data. This should be fixed in the next release.
- All matrices are row-major only. Currently, column-major matrices are not supported.
- At most, one binding semantic per uniform variable is supported by the compiler. Using multiple binding semantics per uniform variable where each semantic is for a different profile is not supported.

Known Issues in the Runtime

- The API entry point **`cgIsParameterReferenced()`** returns **`true`** even if a parameter may not be referenced in the final compiled output.
- In some cases, setting a parameter that is declared in a Cg program but never referenced can cause the runtime to return an error.

Known Issues in the ARB Fragment Program Profile

- Accessing OpenGL state structure similar to the ARB vertex program profile of Cg is not yet supported. This limitation can be somewhat inefficiently overcome by setting explicit uniform parameters to OpenGL state in an application.
- This profile is still in a developmental stage because very limited implementations of the ARB fragment program in OpenGL have been available.

Known Issue in the ps_2_* Profiles

Writing to multiple color outputs is not yet supported.

Known Issues in the arbfp1, fp30, and ps_2_* Profiles

- Conditional assignments to array elements and assignments in **`if/else`** blocks to array elements do not work in all cases.
- The **`%`** operator is not supported. Integer division is not fully emulated and is implemented as floating-point division

Known Issue in the fp20 Profile

The **`FOG`** varying input semantic is not yet supported in this profile

Known Issues in the ps_1_* and fp20 Profiles

Because the underlying hardware support for the **`fp20`** and **`ps_1_*`** profiles is very limited and rigid, it is not always possible to compile some seemingly simple Cg programs in these profiles. To learn more about the limitations of these profiles, please read the presentations available at

http://developer.nvidia.com/view.asp?IO=gdc2001_texture_shaders

http://developer.nvidia.com/view.asp?IO=gdc2001_programmable_texture

For more details, please read the NV_register_combiners and NV_texture_shader OpenGL extensions, or the DirectX PS 1.1, PS 1.2, and PS 1.3 pixel shader specifications.

Known Issue in the Standard Library

The **`noise()`** family of functions in the Standard Library is not yet supported.

Other Do’s and Don'ts for Using the Compiler

- Specify binding semantics for all varying data. Leaving it up to the compiler to allocate resources to varying data with unspecified binding semantics can cause unexpected results. One complicating factor is that it is legal to have multiple input varying variables with the same binding semantic. Hence, it is not always easy for the compiler to allocate resources the way you might expect
- Although partial writes to outputs are allowed in profiles that support them, they are not recommended.
- Avoid using unwritten **`out`** parameters in the program.