Technical Brief

NVIDIA Unified Compiler Technology

Unleashing Innovations

Getting the Most Out of the Hardware

NVIDIA graphics architectures have come a long way since the company's first graphics processing units (GPUs). Today, the NVIDIA® GeForce™ FX family of Microsoft® DirectX® 9.0 GPUs offers developers the ability to create sophisticated visual effects in real time. And with a level of image quality that is rapidly closing the gap between real-time computer-generated (CG) graphics and offline rendering of the film world. Developers are able to develop long shaders with minimal restrictions using DirectX 9 pixel shader version 2.0+ and OpenGL fragment programs. This enables complex effects to be implemented in pixel shaders and allows developers to generate photorealistic effects in real time.

Delivering industry-leading graphics solutions entails a broad set of challenges and even some fortune telling. Hardware designers not only must continually push the performance and functionality forward, but also anticipate the future direction for the major software application programming interfaces (APIs). Even with attention to every detail, coupling a new architecture with the long list of emerging application requirements from the various APIs can be daunting. When a new GPU is released, its new architecture may not suit the latest software programming techniques for one API, yet it may be ideally suited for the programming techniques of another.

The new NVIDIA unified compiler technology overcomes these challenges and provides a new level of freedom for hardware innovations. Just as computer system programming language compilers provide software developers with optimized application performance regardless of the latest API requirements, the NVIDIA unified compiler technology maximizes performance for the combination of the latest APIs and NVIDIA hardware. Instead of time-consuming and sometimes disappointing hand-tuning efforts, the NVIDIA unified compiler technology ensures that each software instruction is automatically executed in a manner that streamlines the execution of graphics applications.

This paper provides an overview of the basic concepts behind the NVIDIA unified compiler technology.

The NVIDIA GeForce FX Architecture

Instruction Translation

The NVIDIA unified compiler technology introduces an additional level of intelligence into NVIDIA drivers. The built-in intelligence fills the role of a compiler by translating higher-level instructions—programming instructions written to an API such as OpenGL or DirectX 9—into lower-level instructions understood by the GPU (Figure 1).

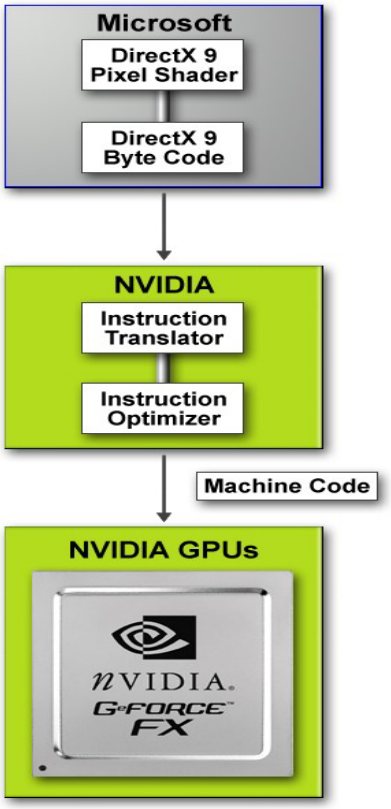

Figure 1. Built into the NVIDIA® ForceWare™ graphics driver, the NVIDIA unified compiler technology translates program instructions into hardware-ready instructions tailored to optimally use NVIDIA hardware

Automatic Tuning

Besides performing this translation process, the NVIDIA unified compiler technology includes numerous cutting-edge matching and rescheduling algorithms. This optimization step ensures that each application program instruction is translated into the least number of GPU instructions possible, and that each GPU instruction takes optimal advantage of the NVIDIA GeForce FX architectures advanced capabilities.

This step is particularly important in a feature-rich, fully programmable architecture such as the GeForce FX family architecture. This architecture has evolved to include many features that go beyond Microsoft DirectX 9 functionality. For example, the GeForce FX GPUs include multiple parallel adders. Written to an API such as DirectX 9, programs that call out multiple addition operations would result in multiple GPU instructions to perform the addition operations. The NVIDIA unified compiler technology, designed to identify operations involving multiple addition steps, can streamline multiple adds into a single instruction and significantly improve application performance.

This step also allows each API to be supported on GeForce FX GPUs with the best possible performance. For example, the latest Microsoft DirectX 9 API uses a programming convention that, without compilation, would slow operation on NVIDIA GPUs. Operands may be passed to the GPU in a specific order—texture, math, texture, math—which is different than what would work best with the GeForce FX hardware design. The NVIDIA unified compiler technology efficiently translates these operands into the order that maximizes execution on NVIDIA GPUs—texture, texture, math, math. This one compiler feature can deliver a 60 percent performance improvement for DirectX 9 applications, and points out how a minor programming difference can result in significant performance impact on programmable GPUs.

Compiler technology tunes DirectX 9 execution on GeForce FX GPUs, and can be used to correct any similar conflict that arises with future APIs. Following are other examples of the automatic tuning performed by the NVIDIA unified compiler.

Example 1

The following input shader takes 11.55 effective cycles due to a 7-pass shader that uses 5 registers:

```
ps_2_0

texld r4, t0, s0       //   0
texld r1, t1, s1       //   0
texld r2, t2, s2       //   1
texld r3, t3, s3       //   1
add r0, v0, r3         //   2
mul r0, r4, r0         //   4
mad oC0, r1, r2, r0    //   6
```

When optimized through the compiler, it takes 5.38 effective cycles due to a 4-pass shader that uses 3 registers. Hence, the output shader runs twice as fast as the input shader. This illustrates how the compiler chooses tex-tex-blend-blend patterns to utilize different units in the GeForce FX shader pipeline.

```
ps_2_0

texld r1, t1, s1    //   0
texld r0, t2, s2    //   0
mulr r0, r1, r0     //   0
texld r1, t3, s3    //   1
texld r2, t0, s0    //   1
addr r1, v1, r1     //   2
mulr r1, r2, r1     //   2
addr r0, r1, r0     //   3
```

Example 2

Here is another example that illustrates different optimizations the compiler does and how it picks instructions based on different criteria. The input shader takes 25.53 effective passes because it is an 11-pass shader that uses 7 registers:

```
ps_2_a

texld r0, t0.xyzz, s2     // 0
texld r1, t4.xyzz, s0     // 1
texld r2, t1.xyzz, s1     // 2
texld r3, t2.xyzz, s3     // 3
mul r3.xyz, r0.w, r0      // 3
mul r3.xyz, r1, r3        // 5
mul r3.xyz, r3, c0        // 6
mad r4.xyz, r2.w, r2, -r3 // 7
mad r5.xyz, r3.w, r4, r3  // 8
mul r6.xyz, r5, c1.x      // 8
mov r6.w, c1.w            // 9
mov oC0, r6               // 10
```

The NVIDIA unified compiler can automatically optimize this program to take 7.06 effective cycles with a 5-pass and 4-register shader. This example also shows how the compiler automatically schedules instructions considering several criteria. It pairs textures only in one case because it has to choose between saving cycles by reducing bank conflicts versus pairing textures.

```
ps_2_a

texld r1,  t0, s2      //   0
texld r0, t2, s3       //   0
mul r0.xyz, r1.w, r1   //   0
texld r3, t4, s0       //   1
mul r0.xyz, r3, r0     //   1
texld r1, t1, s1       //   2
mul r1.xyz, r1.w, r1   //   2
mul r0.xyz, r0, c0     //   3
add r1.xyz, -r0, r1    //   3
mul r1.xyz, r0.w, r1   //   3
add r0.xyz, r0, r1     //   4
mul r0.xyz, r0, c1.x   //   4
mov r0.w, c1.w         //   4
```

Example 3

Here is another example where the compiler minimizes the number of registers to achieve good performance. The input shader program runs in 9.38 effective cycles:

```
ps_2_0

texld r2, t0, s0       // 0
texld r3, t1, s1       // 0
mad r1, v1, c1.x, c1.y // 1
dp3 r1, r2, r1         // 1
add r0, r1, c0         // 2
mul r1, v0, r0         // 3
mul r0.xyz, r1, r3     // 5
mov r0.w, r3           // 6
mov oC0 r0             // 7
```

The optimal output program generated by the compiler runs in 5 effective cycles:

```
ps_2_0

texld r0, t0, s0      //   0
mul r1.xyz, v1, c1.x  //   1
add r1.xyz, c1.y, r1  //   1
dp3 r0.x, r0, r1      //   1
add r0.xyz, r0.x, c0  //   2
mul r0.xyz, v0, r0    //   3
texld r1, t1, s1      //   4
mul r0.xyz, r0, r1    //   4
mov r0.w, r1          //   4
```

The Benefits of a Graphics Compiler

For Programmers

Programmers can benefit from these features:

- **Fast access to the latest hardware advances.** Compilers automatically leverage the feature-rich GeForce FX platform without requiring developers to know about the latest implementation details. The NVIDIA unified compiler automates the tuning of an application to the hardware.
- **Shortened development cycles and optimized runtime performance.** Because the technology carries out the tuning at runtime, programmers gain the time savings of writing a single version of each application. As long as application developers write to an industry-standard API, their applications will run with maximized efficiency on the GeForce FX GPUs.

- **Transparent benefit from NVIDIA enhancements and special features**. Even if the application is written to an API in a manner to maximize portability among platforms, the NVIDIA compiler will take advantage of NVIDIA hardware that goes beyond the API. The three previous examples illustrate this.
- **Forward and backward compatibility.** Today's applications ensure that tomorrows drivers will optimize the same code to best fit tomorrow's GPUs.

For Users

The NVIDIA unified compiler technology does not compromise image quality in any way. The matching and rescheduling algorithms make no compromises to image quality in order to achieve the accelerated execution results.

Unlike other tuning approaches that sometimes require tradeoffs, compiler technology introduces a way to inject intelligent tuning and gain performance benefits without reducing anisotropic filtering, trilinear filtering, antialiasing, or other parameters that affect the overall visual experience of the users. Users will see performance gains in increased interactivity for games and other real-time applications.

For the Industry

Compiler technology will serve as a positive force for graphics applications. The key benefits for the entire industry include

- **Innovations**. Hardware designers are not forced to evolve hardware architectures in step with APIs. This freedom fosters innovation while ensuring compatibility with APIs and smooth migration paths for developers and users. Users that demand the best graphics capabilities and performance will not be held back by hardware designers who divert resources to tune to a particular API.
- **Performance**. The compiler component of the overall graphics architecture opens many opportunities for higher-level tuning efforts. Just as they do for computer systems, compilers can account for more performance advances than either software-only or hardware-only efforts. As the unique link between software and hardware, compilers can implement the latest optimizing algorithms and techniques. This frees developers from having to optimize for a single platform, allowing them to remain focused on their core competencies.
- **Investment protection**. Risks are minimized because hardware architects do not have to accurately predict where APIs will be years down the road. Hardware evolution takes place over years, whereas software can change rapidly and leave hardware designers with difficult issues to overcome if they have not anticipated the change. The result will be graphics solutions that maximize performance for all APIs.

Historical Perspective

Compilers are not new. Computer systems have relied on compiler technology for decades. As machine architectures became more and more complex, the low-level machine languages became too cumbersome to use, and tuning applications became an intensive exercise that required in-depth expertise. Today, application programmers have a host of higher-level programming languages to choose from, and generations of compiler technologies have automated many previously tedious tuning steps.

The same evolutionary forces are at work in the graphics industry. GPUs are significantly more complex today, with many more functions built into the hardware.

With these technologies, NVIDIA has made numerous steps to simplifying programming:

- **The NVIDIA Unified Driver Architecture (UDA)**. A single driver supports all NVIDIA GPUs, shielding the programmer and users from the various differences in the graphics solutions, while taking advantage of whatever hardware features are available.
- **The NVIDIA unified compiler technology**. The NVIDIA unified compiler technology automates the optimization of graphics applications written for various APIs. It accomplishes this by matching and rescheduling instructions to execute in the shortest possible number of instructions on NVIDIA GPUs.

Summary

With the NVIDIA unified compiler technology, NVIDIA continues a tradition of driving architectural enhancements that foster innovation and high performance. By designing graphics solutions that solve the most challenging problems and deliver the highest levels of programmability, NVIDIA offers uncompromised graphics power and unmatched visual experiences to numerous applications. The NVIDIA unified compiler gives NVIDIA engineers the freedom that they need, and avoids having to customize an architecture to any one version of an API—an architecture that would be hard to evolve for other APIs and future changes of direction.

In the long term, striving for the most feature-rich architecture will push the industry forward much more aggressively. NVIDIA remains committed to this approach, and will continue to aim for far-reaching milestones marking both hardware and software advancements.

Notice

Trademarks

Copyright

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