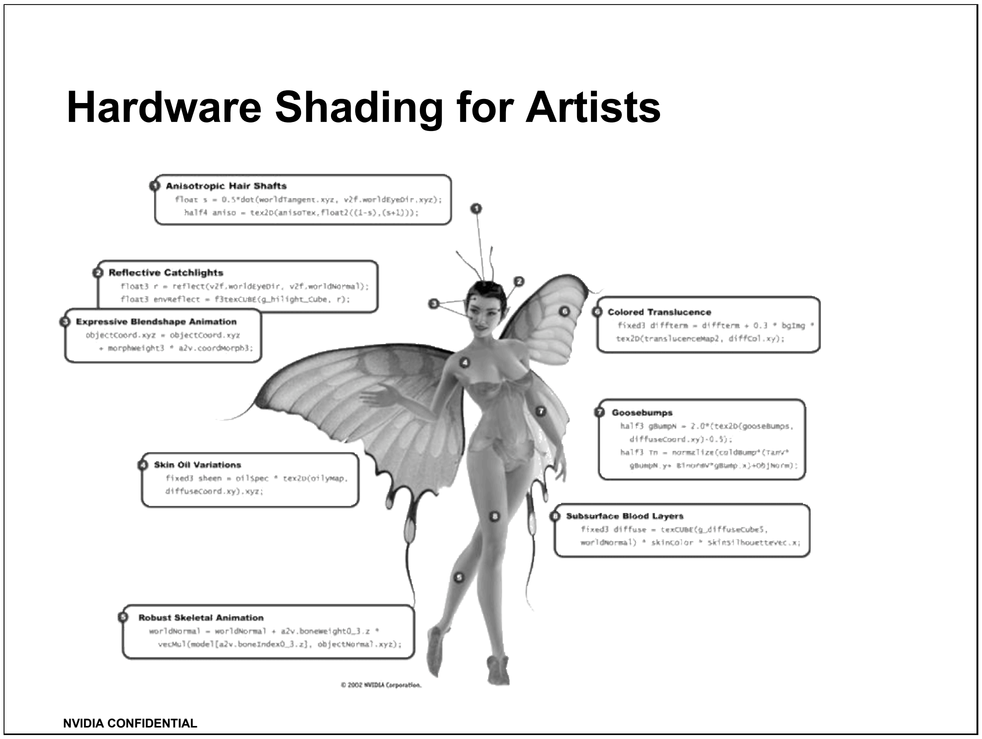


Hardware Shading for Artists

1 **Anisotropic Hair Shafts**

```
float s = 0.5*dot(worldTangent.xyz, v2f.worldEyeDir.xyz);
  half4 aniso = tex2D(anisoTex,float2((1-s),(s+1)));
```

2 **Reflective Catchlights**

```
float3 r = reflect(v2f.worldEyeDir, v2f.worldNormal);
float3 envReflect = f3texCUBE(g_hilight_Cube, r);
```

3 **Expressive Blendshape Animation**

```
objectCoord.xyz = objectCoord.xyz
  + morphWeight3 * a2v.coordMorph3;
```

4 **Skin Oil Variations**

```
fixed3 sheen = oilSpec * tex2D(oilyMap,
diffuseCoord.xy).xyz;
```

5 **Robust Skeletal Animation**

```
worldNormal = worldNormal + a2v.boneWeight0_3.z *
  vecMul(model[a2v.boneIndex0_3.z], objectNormal.xyz);
```

6 **Colored Translucence**

```
fixed3 diffterm = diffterm + 0.3 * bgImg *
tex2D(translucenceMap2, diffCol.xy);
```

7 **Goosebumps**

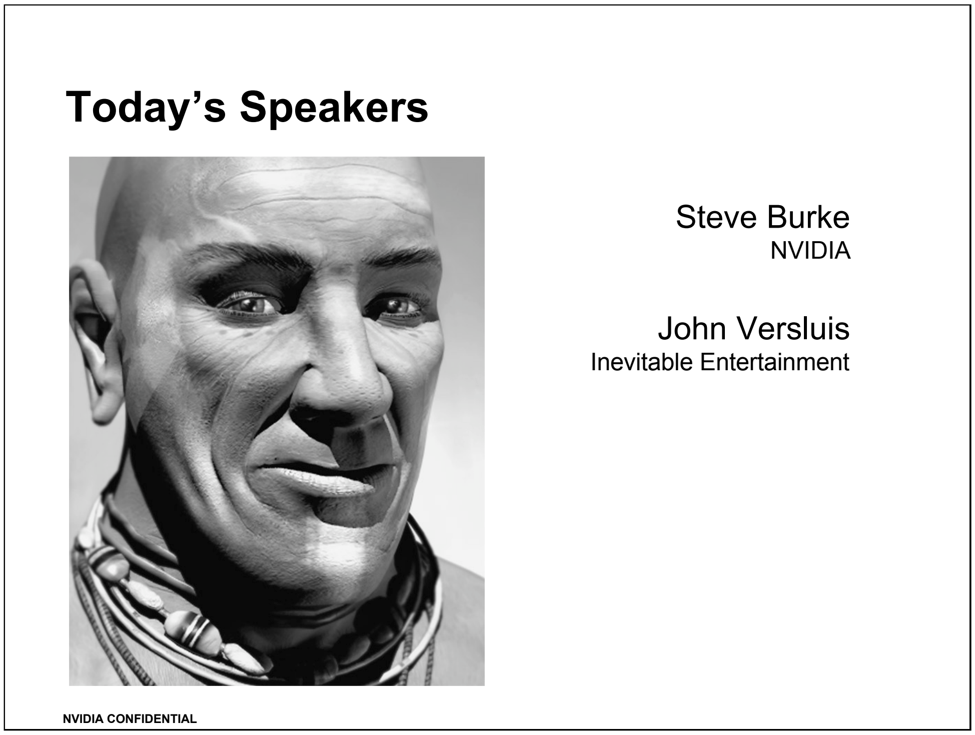
```
half3 gBumpN = 2.0*(tex2D(gooseBumps,
  diffuseCoord.xy)-0.5);
half3 Tn = normalize(coldBump*(TanV*
  gBumpN.y+ BinormV*gBump.x)+ObjNorm);
```

8 **Subsurface Blood Layers**

```
fixed3 diffuse = texCUBE(g_diffuseCube5,
worldNormal) * skinColor * SkinSilhouetteVec.x;
```

NVIDIA CONFIDENTIAL

Just an image to remind us of the power of programmable shaders.



Today's Speakers

Steve Burke
NVIDIA

John Versluis
Inevitable Entertainment

NVIDIA CONFIDENTIAL

Experience in game industry, high-end 3D art, blah, blah, blah,

Interest in real-time work and role at NVIDIA, working with developers to raise the overall level of quality in real-time art by providing both technical information and artistic information to artists.

John's experience in games and high-end 3d art. Expertise at technical art and real-time game issues.

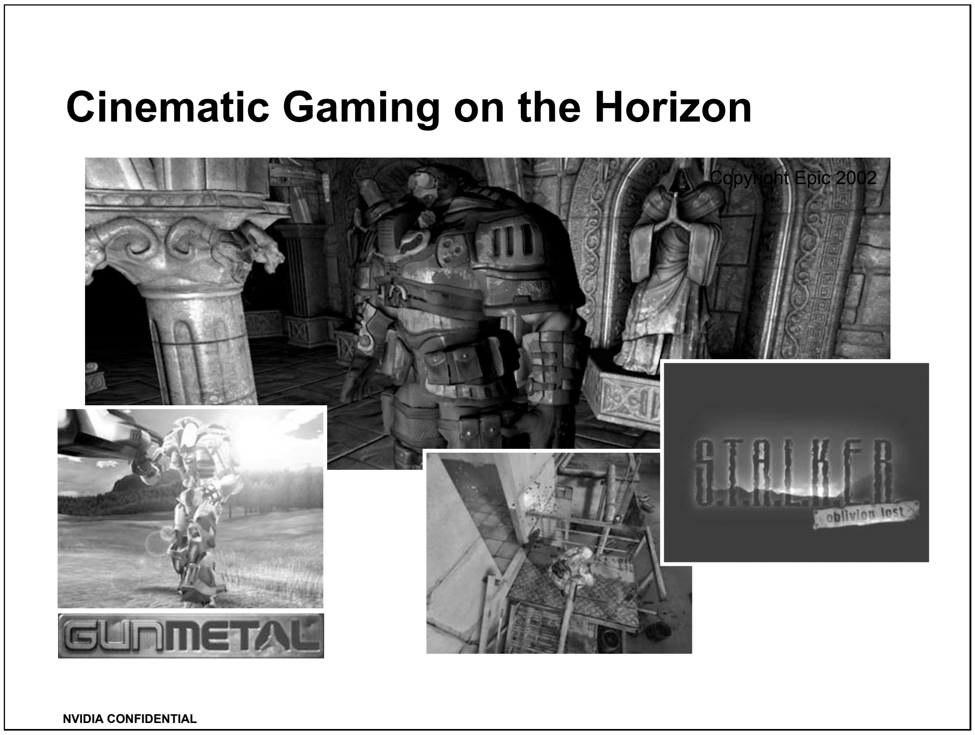
Hardware Shaders in Games

Copyright Epic 2002

Hardware Shaders Bring Your Game Closer to Cinematic Quality

Discuss some of the current games using hardware shaders, effects on gameplay, quality of user experience.

Ability for artist to be more expressive. Effects also. Not just hardware shaders.



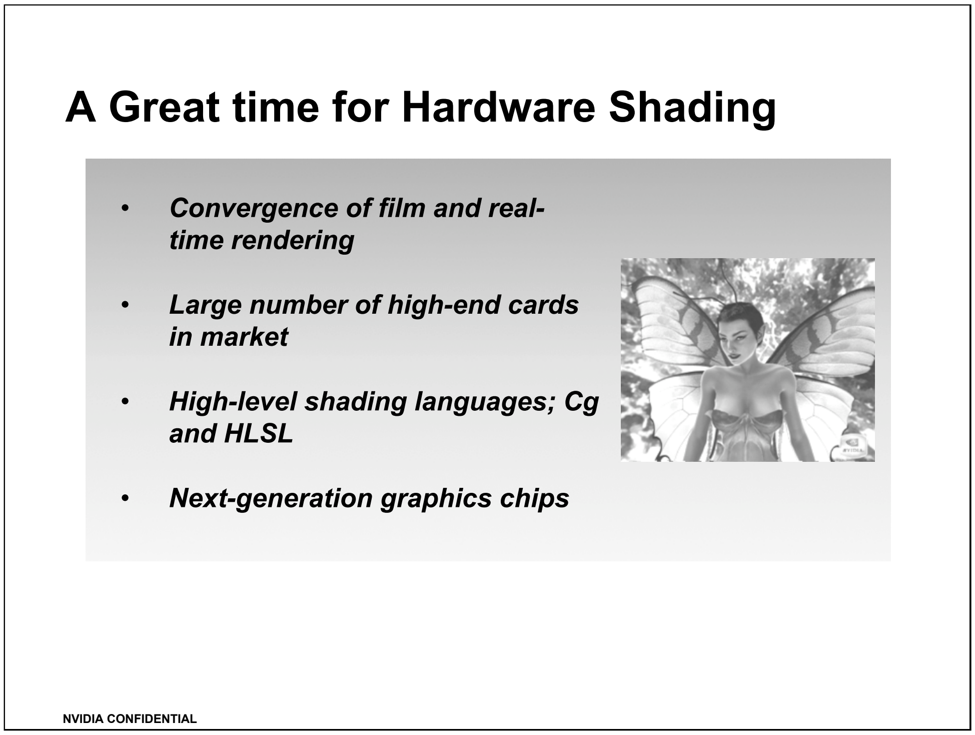
Cinematic Gaming on the Horizon

Copyright Epic 2002

NVIDIA CONFIDENTIAL

Discuss some of the current games using hardware shaders, effects on gameplay, quality of user experience.

Ability for artist to be more expressive. Effects also. Not just hardware shaders.

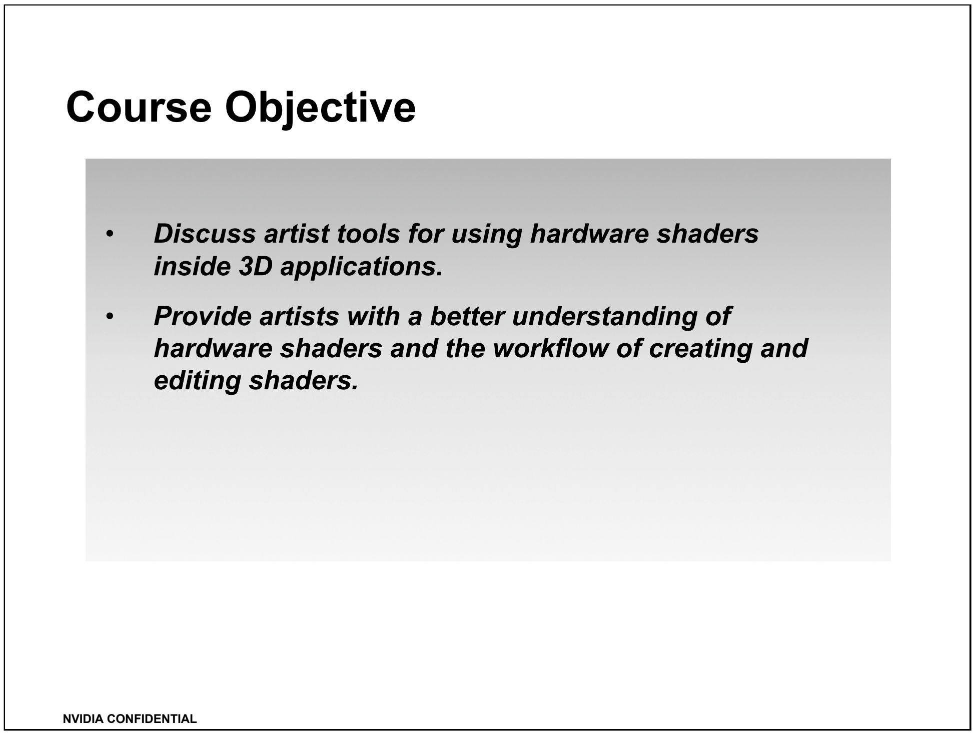


A Great time for Hardware Shading

- ***Convergence of film and real-time rendering***
- ***Large number of high-end cards in market***
- ***High-level shading languages; Cg and HLSL***
- ***Next-generation graphics chips***

There have been a lot of changes in real-time 3D in the last year. Now, more than ever, it is practical and profitable to support high-end vertex and pixel shaders in your game.

CPUs getting more and more use with fun stuff like physics calculations and animation



Course Objective

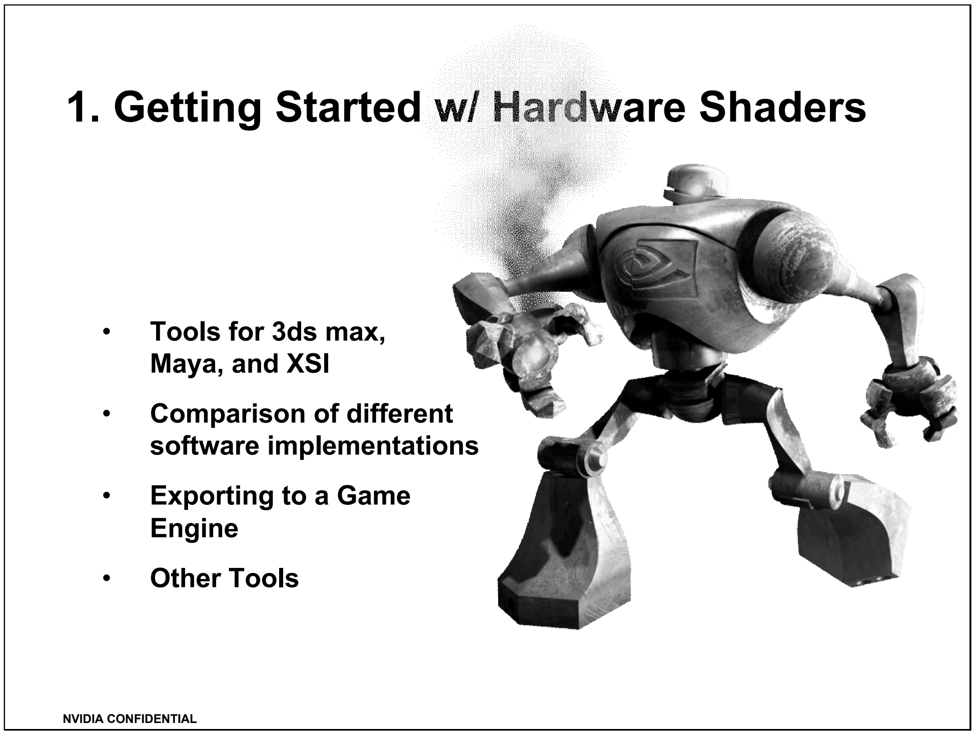
- ***Discuss artist tools for using hardware shaders inside 3D applications.***
- ***Provide artists with a better understanding of hardware shaders and the workflow of creating and editing shaders.***

I want to show the artists the tools involved in hardware shader design but also to provide a solid understanding of what hardware shaders are, how they work and how artists can use them to their advantage.

There are substantial differences between how shaders normally work in a 3D program and how they work in real-time. I want to point out these differences and basically give artists they need to use the tools.

Important for artist to know constraints. Seeing what is possible allows you to plan better what type of effects you want to create.

First, give a qiick overview of the workflow and the tools invloved. Second, talk about the shaders and all the nuances of working with hardware shaders as oppossed to software shaders.



1. Getting Started w/ Hardware Shaders

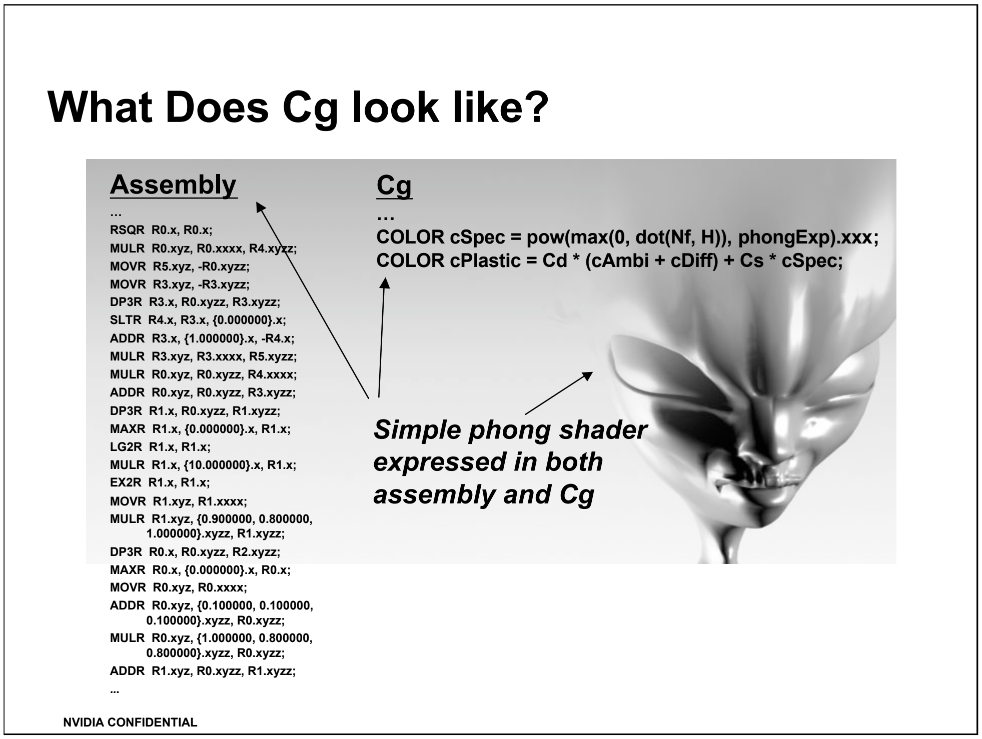
- Tools for 3ds max, Maya, and XSI
- Comparison of different software implementations
- Exporting to a Game Engine
- Other Tools

NVIDIA CONFIDENTIAL

Cg is one of the big factors in why you can now work with hardware shaders inside your favorite 3D programs.

Cg allows for shader writing in a painless way. Also have NVB Exporter and CgFX Viewer but they aren’t necessary

Other tools can support Cg or CgFX



What Does Cg look like?

Assembly

```
…
RSQR  R0.x, R0.x;
MULR  R0.xyz, R0.xxxx, R4.xyzz;
MOVR  R5.xyz, -R0.xyzz;
MOVR  R3.xyz, -R3.xyzz;
DP3R  R3.x, R0.xyzz, R3.xyzz;
SLTR  R4.x, R3.x, {0.000000}.x;
ADDR  R3.x, {1.000000}.x, -R4.x;
MULR  R3.xyz, R3.xxxx, R5.xyzz;
MULR  R0.xyz, R0.xyzz, R4.xxxx;
ADDR  R0.xyz, R0.xyzz, R3.xyzz;
DP3R  R1.x, R0.xyzz, R1.xyzz;
MAXR  R1.x, {0.000000}.x, R1.x;
LG2R  R1.x, R1.x;
MULR  R1.x, {10.000000}.x, R1.x;
EX2R  R1.x, R1.x;
MOVR  R1.xyz, R1.xxxx;
MULR  R1.xyz, {0.900000, 0.800000,
        1.000000}.xyzz, R1.xyzz;
DP3R  R0.x, R0.xyzz, R2.xyzz;
MAXR  R0.x, {0.000000}.x, R0.x;
MOVR  R0.xyz, R0.xxxx;
ADDR  R0.xyz, {0.100000, 0.100000,
        0.100000}.xyzz, R0.xyzz;
MULR  R0.xyz, {1.000000, 0.800000,
        0.800000}.xyzz, R0.xyzz;
ADDR  R1.xyz, R0.xyzz, R1.xyzz;
...
```

Cg

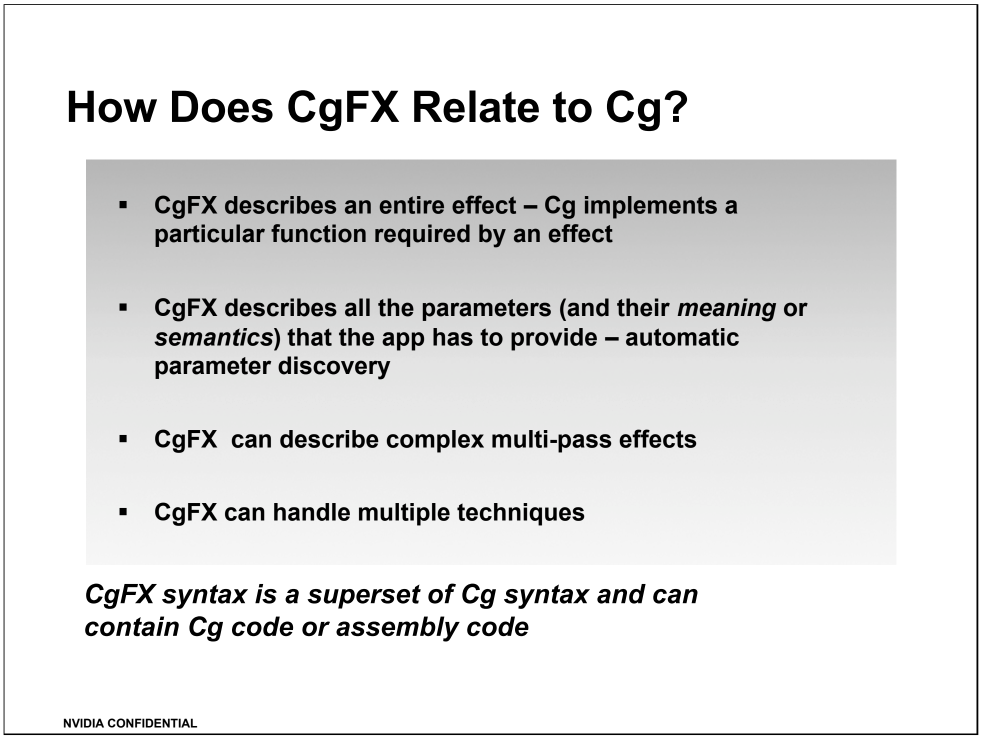
```
…
COLOR cSpec = pow(max(0, dot(Nf, H)), phongExp).xxx;
COLOR cPlastic = Cd * (cAmbi + cDiff) + Cs * cSpec;
```

Simple phong shader expressed in both assembly and Cg

NVIDIA CONFIDENTIAL

Cg, is infinitely more understandable to the programmer – they clearly can see elements like a specular highlight, a combination of ambient and diffuse colors… But they'll still see some GPU functions – such as a dot product – explicitly addressed in the code. It's a high-level language that works they way hardware rendering does.

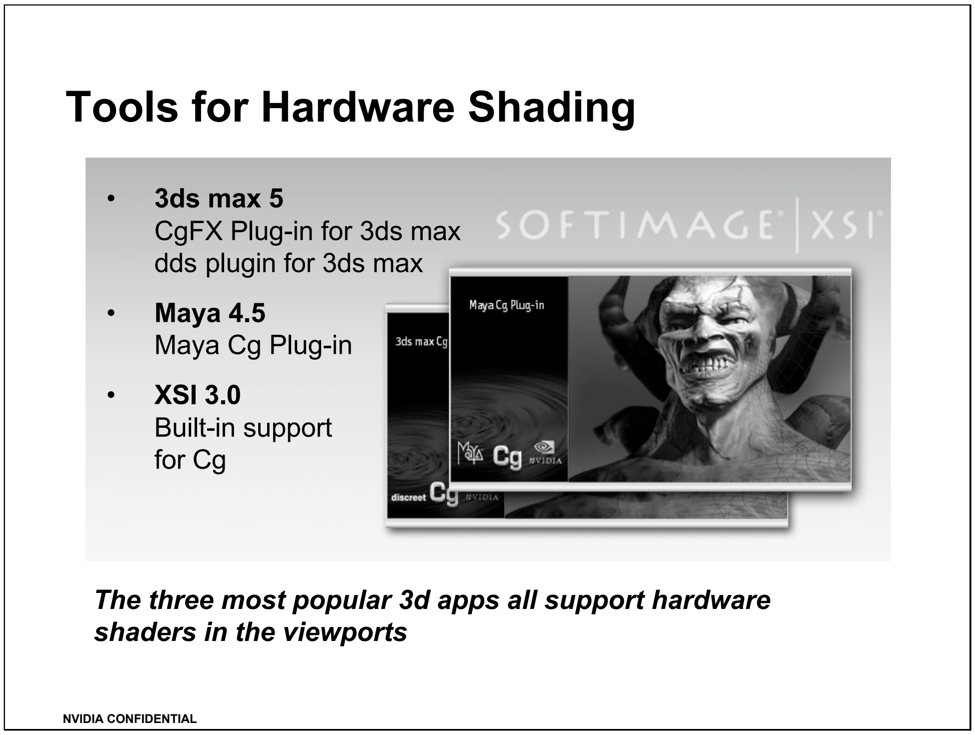
Renderman offers the a higher level of abstraction than Cg but doesn't correlate to the hardware.



How Does CgFX Relate to Cg?

- CgFX describes an entire effect – Cg implements a particular function required by an effect
- CgFX describes all the parameters (and their *meaning* or *semantics*) that the app has to provide – automatic parameter discovery
- CgFX can describe complex multi-pass effects
- CgFX can handle multiple techniques

CgFX syntax is a superset of Cg syntax and can contain Cg code or assembly code



Tools for Hardware Shading

- **3ds max 5**
  CgFX Plug-in for 3ds max
  dds plugin for 3ds max
- **Maya 4.5**
  Maya Cg Plug-in
- **XSI 3.0**
  Built-in support
  for Cg

SOFTIMAGE | XSI
Maya Cg Plug-in
3ds max Cg
Maya
Cg
NVIDIA
discreet
Cg
NVIDIA

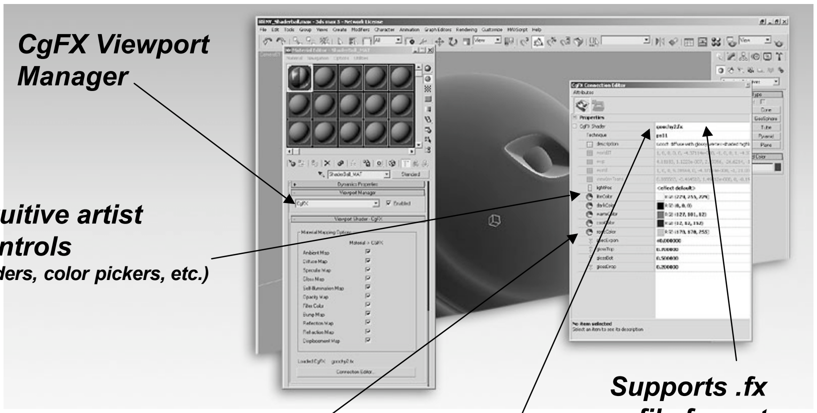
The three most popular 3d apps all support hardware shaders in the viewports

Cg is one of the big factors in why you can now work with hardware shaders inside your favorite 3D programs.

Cg allows for shader writing in a painless way. Also have NVB Exporter and CgFX Viewer but they aren’t necessary

Other tools can support Cg or CgFX. Need exporter to bring tweaked shaders into your game engine.

Cg implementation: 3ds max 5



CgFX Viewport Manager

Intuitive artist controls
(sliders, color pickers, etc.)

Supports .fx file format

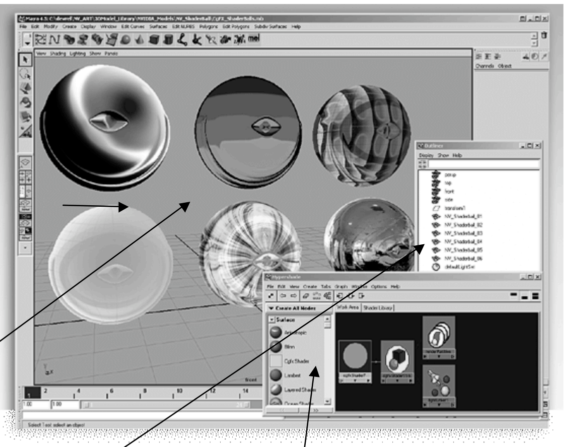
Dynamic, shader-specific GUI

Multiple Techniques for fallbacks

NVIDIA CONFIDENTIAL

Finally, here’s the latest code in action, showing a bunch of different CgFX shaders right in max, alongside a game engine using the same shaders. We’re showing this off all day, every day at the NVIDIA booth.

Cg implementation: Maya 4.5



Supports .fx file format

Intuitive, shader-specific, artist controls

Slider control over key real-time parameters (e.g., bump depth)

Sample shaders include:

Bumpy Shiny, Toon, Anisotropic Metal, Ghostly, Refraction Dispersion, Rainbow

Integrated with Maya's lights

CgFX integrated with Maya's Hypershade

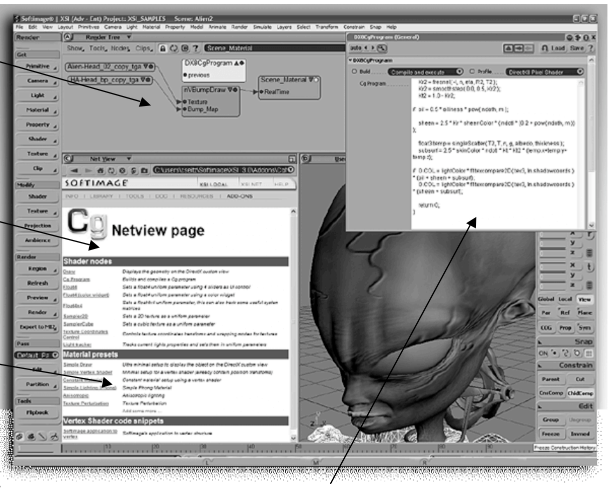
NVIDIA CONFIDENTIAL

Cg implementation: Softimage|XSI 3.0

Cg Integration in XSI's Render Tree

Net View for help, samples & documentation

Interactive shader builder



Shader nodes

Draw	Displays the geometry on the DirectX custom view
Cg Program	Builds and compiles a Cg program
Float4	Sets a float4 uniform parameter using 4 sliders as UI control
Float4 (color widget)	Sets a float4 uniform parameter using a color widget
Float4x4	Sets a float4x4 uniform parameter, this can also track some useful system matrices
Sampler2D	Sets a 2D texture as a uniform parameter
SamplerCube	Sets a cubic texture as a uniform parameter
Texture Coordinates Control	Controls texture coordinates transforms and wrapping modes for textures
Light tracker	Tracks current lights properties and sets them in uniform parameters

Material presets

Simple Draw	Ultra minimal setup to display the object on the DirectX custom view
Simple Vertex Shader	Minimal setup for a vertex shader (already contain position transforms)
Constant Material	Constant material setup using a vertex shader
Simple Lighting (Phong)	Simple Phong Material
Anisotropic	Anisotropic lighting
Texture Perturbation	Texture Perturbation
	Add some more ...

Vertex Shader code snippets

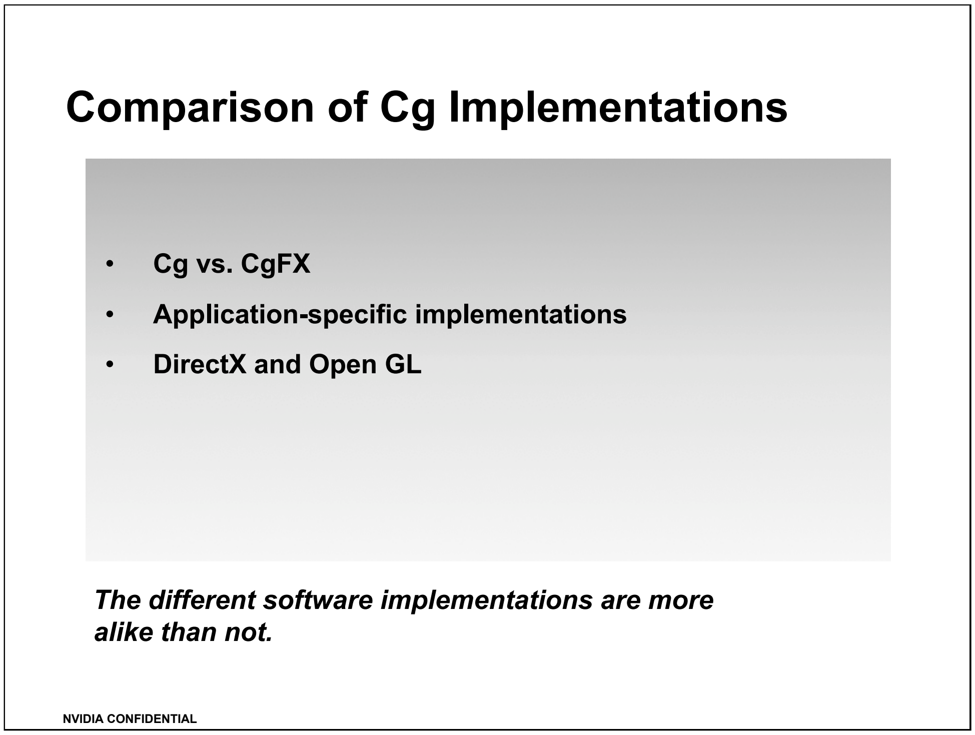
Softimage application to vertex	Softimage's application to vertex structure

```
    Kr2 = fresnel(-I, n, eta, R2, T2);
    Kr2 = smoothstep(0.0, 0.5, Kr2);
    Kt2 = 1.0 - Kr2;

//  oil = 0.5 * oiliness * pow(ndoth, m);

    sheen = 2.5 * Kr * sheenColor * (ndotl * (0.2 + pow(ndoth, m))
);

    float3 temp = singleScatter(T2, T, n, g, albedo, thickness);
    subsurf = 2.5 * skinColor * ndotl * Kt * Kt2 * (temp.x+temp.y+
temp.z);

//  O.COL = lightColor * f1texcompare2D(tex3, In.shadowcoords )
* (oil + sheen + subsurf);
    O.COL = lightColor * f1texcompare2D(tex3, In.shadowcoords )
* (sheen + subsurf);

    return O;
}
```

Shipping with XSI 3.0

Direct Cg code editing and compilation

NVIDIA CONFIDENTIAL



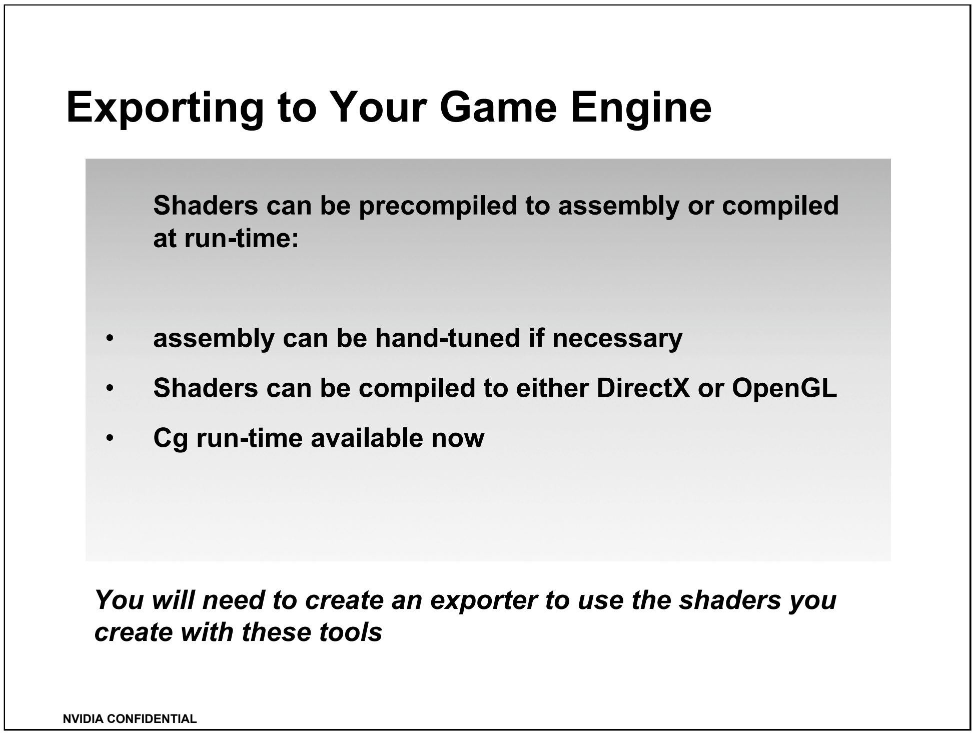
Comparison of Cg Implementations

- Cg vs. CgFX
- Application-specific implementations
- DirectX and Open GL

The different software implementations are more alike than not.

NVIDIA CONFIDENTIAL

Discuss similarities and differences among the different programs. Cg vs.



Exporting to Your Game Engine

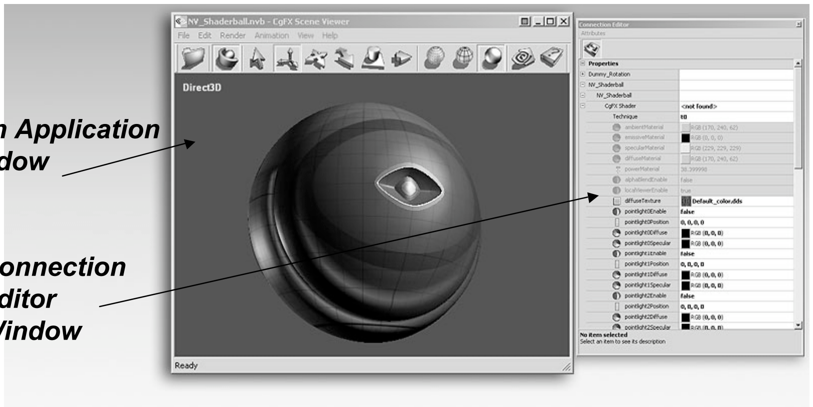
Shaders can be precompiled to assembly or compiled at run-time:

- **assembly can be hand-tuned if necessary**
- **Shaders can be compiled to either DirectX or OpenGL**
- **Cg run-time available now**

You will need to create an exporter to use the shaders you create with these tools

Discuss tradeoff for run-time versus pre-compile

CgFX Viewer


NV_Shaderball.nvb - CgFX Scene Viewer
File Edit Render Animation View Help
Direct3D
Ready
Connection Editor
Attributes

Main Application Window

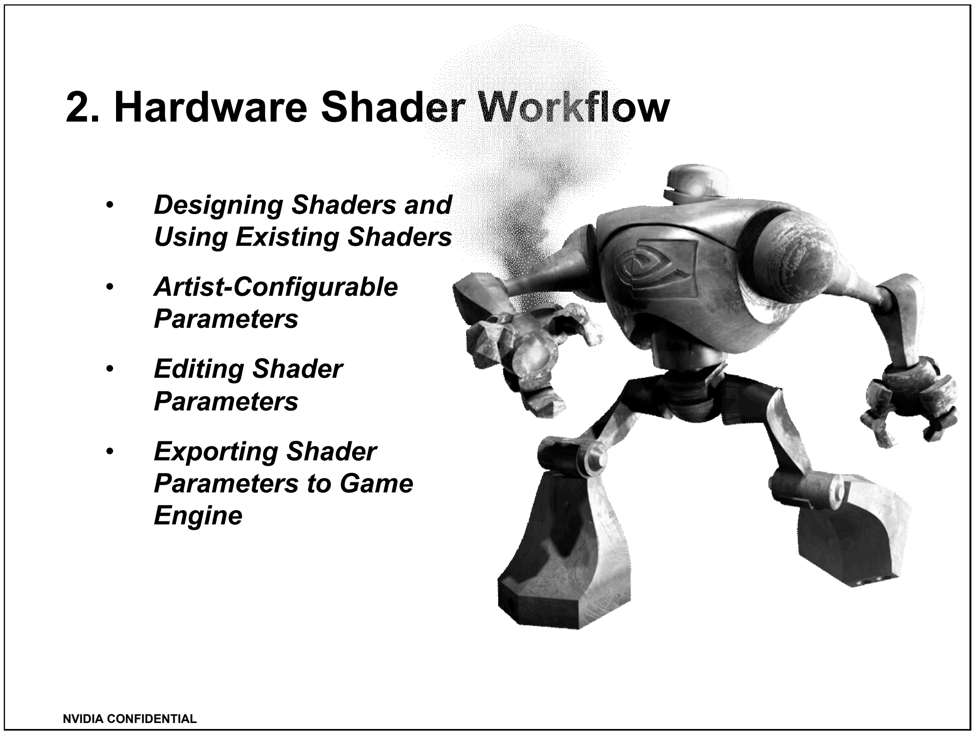
Connection Editor Window

Properties	
Dummy_Rotation	
NV_Shaderball	
NV_Shaderball	
CgFX Shader	<not found>
Technique	t0
ambientMaterial	RGB (170, 240, 62)
emissiveMaterial	RGB (0, 0, 0)
specularMaterial	RGB (229, 229, 229)
diffuseMaterial	RGB (170, 240, 62)
powerMaterial	38.399998
alphaBlendEnable	false
localViewerEnable	true
diffuseTexture	Default_color.dds
pointlight0Enable	false
pointlight0Position	0, 0, 0, 0
pointlight0Diffuse	RGB (0, 0, 0)
pointlight0Specular	RGB (0, 0, 0)
pointlight1Enable	false
pointlight1Position	0, 0, 0, 0
pointlight1Diffuse	RGB (0, 0, 0)
pointlight1Specular	RGB (0, 0, 0)
pointlight2Enable	false
pointlight2Position	0, 0, 0, 0
pointlight2Diffuse	RGB (0, 0, 0)
pointlight2Specular	RGB (0, 0, 0)

No item selected
Select an item to see its description

The CgFX Viewer can be used as a production resource and a code example for implementing CgFX

NVIDIA CONFIDENTIAL



2. Hardware Shader Workflow

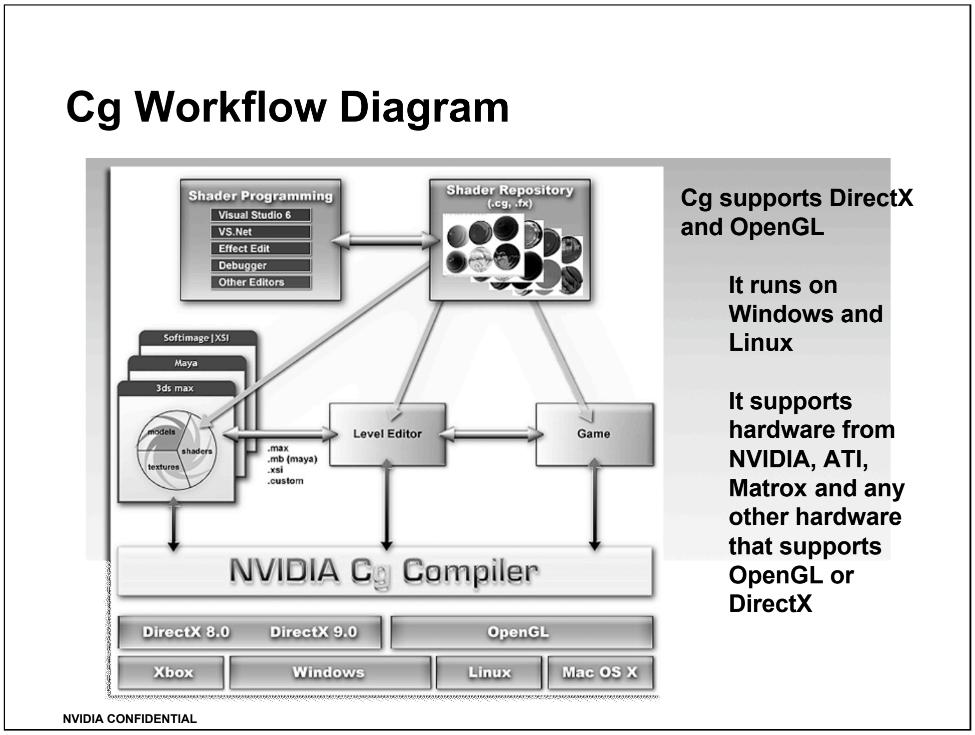
- ***Designing Shaders and Using Existing Shaders***
- ***Artist-Configurable Parameters***
- ***Editing Shader Parameters***
- ***Exporting Shader Parameters to Game Engine***

NVIDIA CONFIDENTIAL

I want to show the artists the tools involved in hardware shader design but also to provide a solid understanding of what hardware shaders are, how they work and how artists can use them to their advantage.

There are substantial differences between how shaders normally work in a 3D program and how they work in real-time. I want to point out these differences and basically give artists they need to use the tools.

Important for artist to know constraints. Seeing what is possible allows you to plan better what type of effects you want to create.



Cg Workflow Diagram

Cg supports DirectX and OpenGL

It runs on Windows and Linux

It supports hardware from NVIDIA, ATI, Matrox and any other hardware that supports OpenGL or DirectX

NVIDIA CONFIDENTIAL

Cg is cross-API and cross-platform.

compiles to either DirectX or OpenGL. That makles Cg cross-platform. These tools run on non-NVIDIA cards as well as NVIDIA cards

Cg can be pre-compiled to assembly or can be built into your game engine.

Create or Acquire Shaders

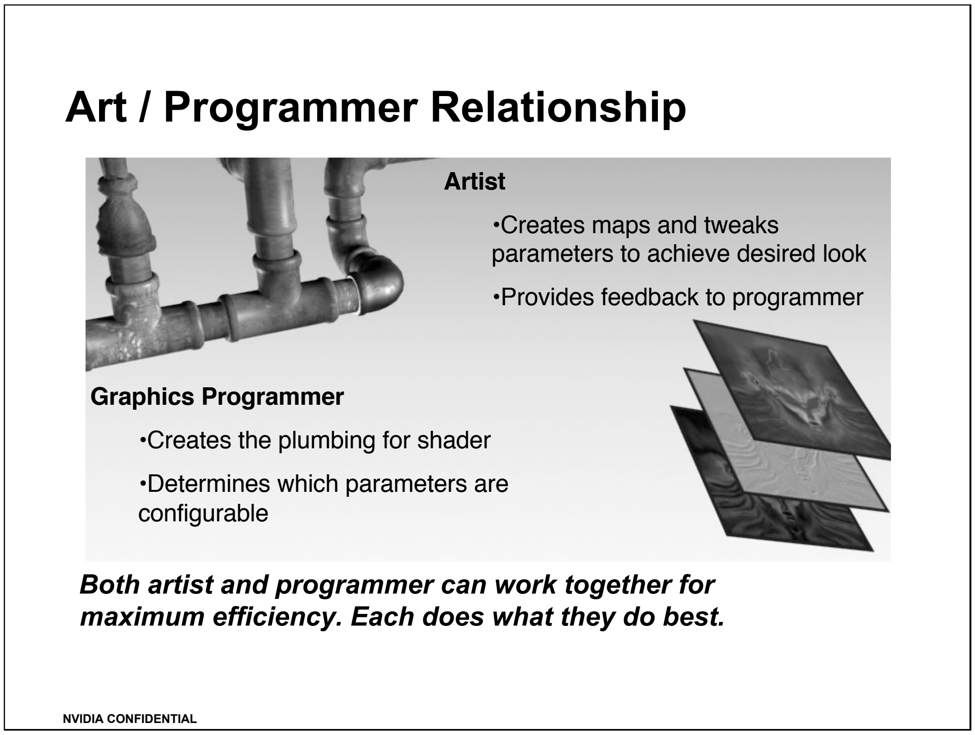
Microsoft Visual C++ - [C:\...\MEDIA\CgFX\AnisoCg.fx *]
File Edit View Insert Project Build BoundsChecker Tools Window Help

```
};

vpconn ani
                                          Trans,
           )
{
    vpconn OUT;

    float3 worldNormal = mul(WorldIT, IN.No
    worldNormal =            (worldNormal);
    //Build f
    float4
    tempPo
    tempPo

    //cor
    float
    float
    //vec
    float
    float                        tPo
    //h =
    float3                          ertToEye
```

- *cgshaders.org*
- *Cg Browser*
- *In-house libraries*

Shaders written in assembly or Cg

NVIDIA CONFIDENTIAL



Art / Programmer Relationship

Artist

- Creates maps and tweaks parameters to achieve desired look
- Provides feedback to programmer

Graphics Programmer

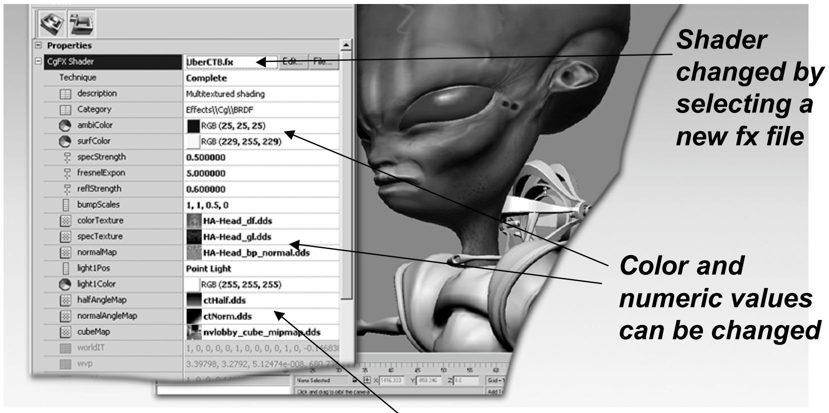
- Creates the plumbing for shader
- Determines which parameters are configurable

Both artist and programmer can work together for maximum efficiency. Each does what they do best.

NVIDIA CONFIDENTIAL

This is the standard way of working and is similar to the way that Pixar and other film companies work. You'll always be better off by having a good relationship with the programming staff.

Customizing Shader Parameters



Properties

CgFX Shader	UberCTB.fx
Technique	Complete
description	Multitextured shading
Category	Effects\\Cg\\BRDF
ambiColor	RGB (25, 25, 25)
surfColor	RGB (229, 255, 229)
specStrength	0.500000
fresnelExpon	5.000000
reflStrength	0.600000
bumpScales	1, 1, 0.5, 0
colorTexture	HA-Head_df.dds
specTexture	HA-Head_gl.dds
normalMap	HA-Head_bp_normal.dds
light1Pos	Point Light
light1Color	RGB (255, 255, 255)
halfAngleMap	ctHalf.dds
normalAngleMap	ctNorm.dds
cubeMap	nvlobby_cube_mipmap.dds
worldIT	1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, -0.146838
wvp	3.39798, 3.2792, 5.12474e-008, 680.77

Shader changed by selecting a new fx file

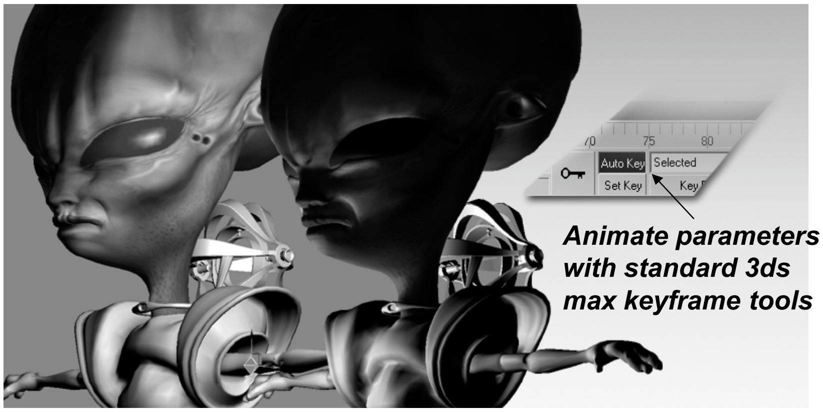
Color and numeric values can be changed

Customizable Parameters are specific to each effect

Bitmaps can be swapped

NVIDIA CONFIDENTIAL

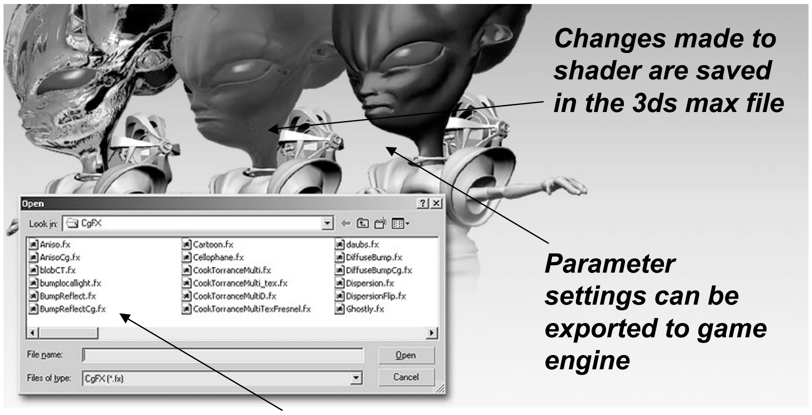
Lights and Animation



Animate parameters with standard 3ds max keyframe tools

Shader reacts to changes in light position

Saving Shader Customizations



Look in: CgFX

File name:

Files of type: CgFX (*.fx)

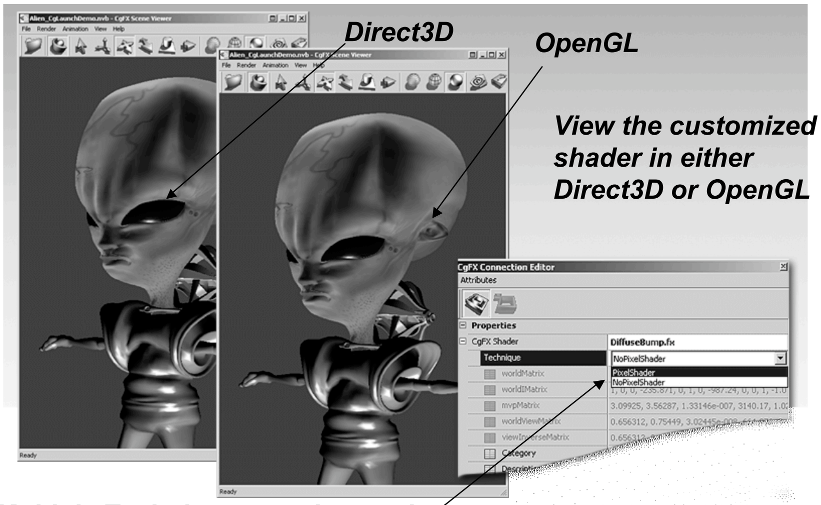
Changes made to shader are saved in the 3ds max file

Parameter settings can be exported to game engine

Loading new shaders is as simple as selecting a new fx file

NVIDIA CONFIDENTIAL

View Shader in Game Engine


Alien_CgLaunchDemo.nvb - CgFX Scene Viewer
File Render Animation View Help
Ready
Alien_CgLaunchDemo.nvb - CgFX Scene Viewer
File Render Animation View Help
Ready
CgFX Connection Editor
Attributes

Properties

CgFX Shader	DiffuseBump.fx
Technique	NoPixelShader PixelShader NoPixelShader
worldMatrix	
worldIMatrix	1, 0, 0, -235.871, 0, 1, 0, -987.24, 0, 0, 1, -1.0
mvpMatrix	3.09925, 3.56287, 1.33146e-007, 3140.17, 1.0
worldViewMatrix	0.656312, 0.75449, 3.02445e-008,
viewInverseMatrix	0.656312,
Category	
Description	

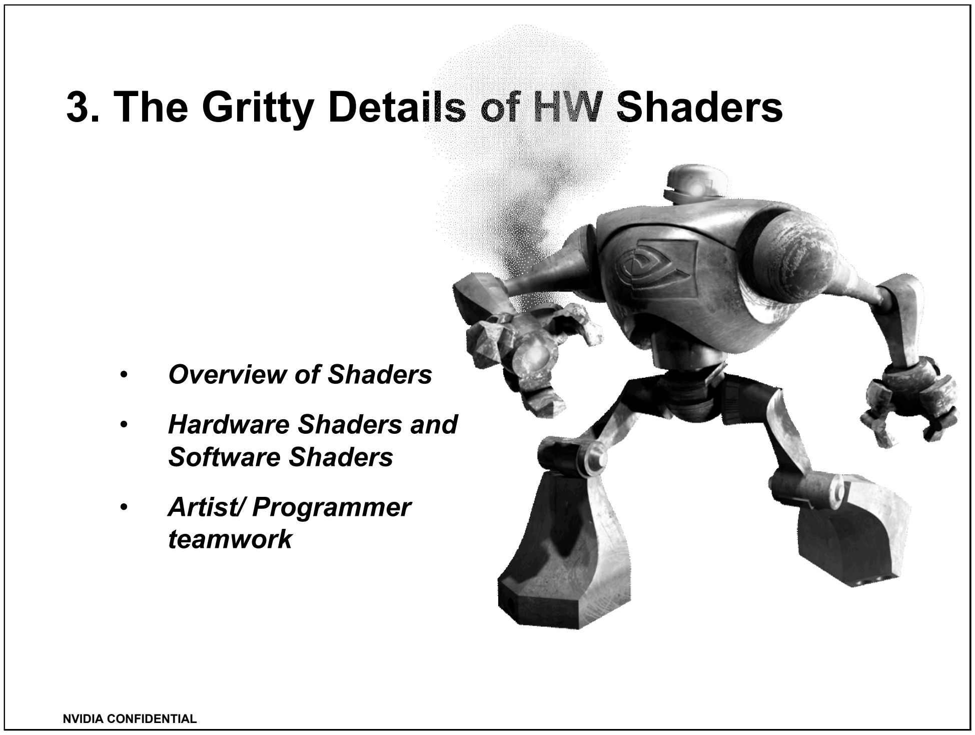
Direct3D

OpenGL

View the customized shader in either Direct3D or OpenGL

Multiple Techniques can be used

NVIDIA CONFIDENTIAL



3. The Gritty Details of HW Shaders

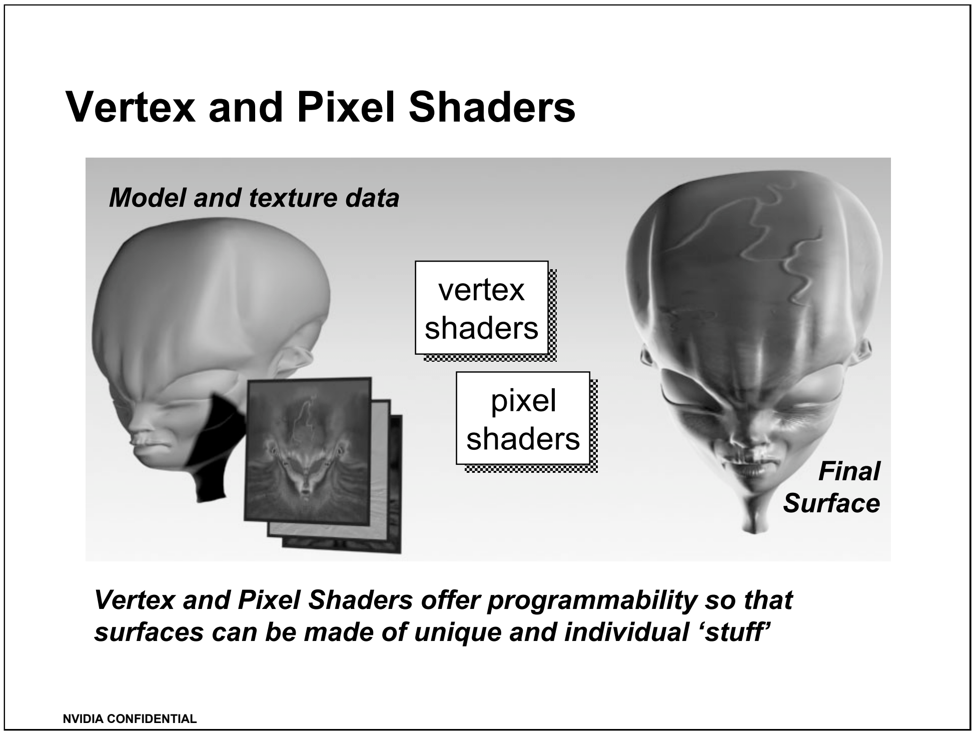
- ***Overview of Shaders***
- ***Hardware Shaders and Software Shaders***
- ***Artist/ Programmer teamwork***

NVIDIA CONFIDENTIAL

Overview of Shaders, be clear about what a shader is.

Big differences between hardware and software shaders.

teamwork. Nobody likes teamwork



Vertex and Pixel Shaders

Model and texture data

vertex shaders

pixel shaders

Final Surface

Vertex and Pixel Shaders offer programmability so that surfaces can be made of unique and individual 'stuff'

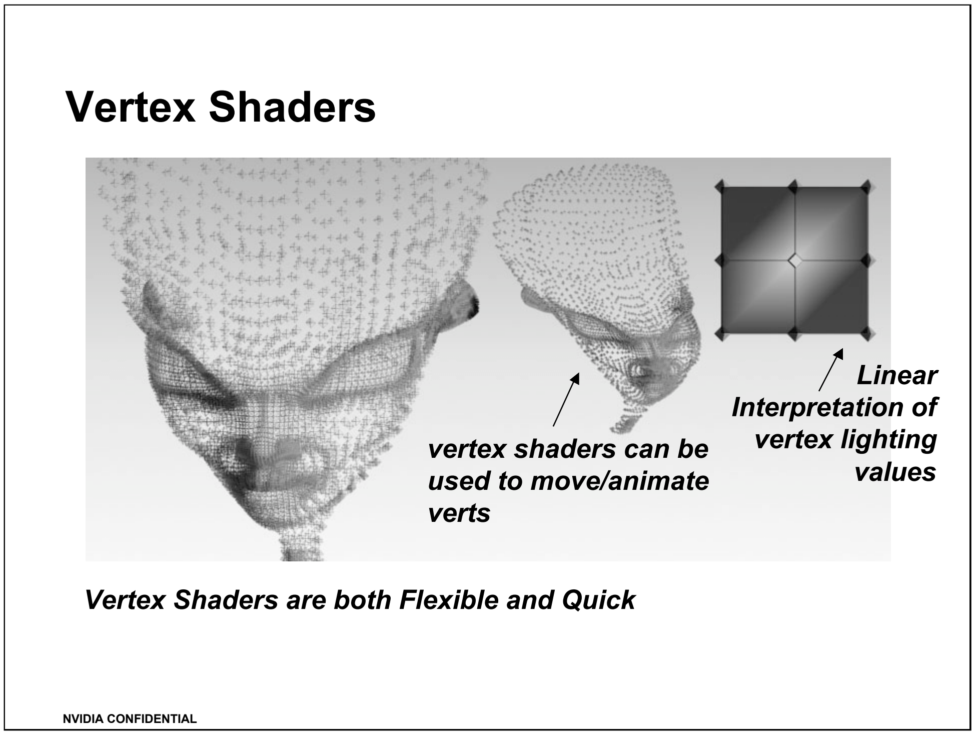
NVIDIA CONFIDENTIAL

Vertex and Pixel shaders offer programmability of surfaces so that the same model and texture data can produce many different looks. Unlike Fixed-function pipeline real-time, not everything is made of the same stuff.

Surface properties like reflectance, light dispersion, shininess, etc. can be programmed on a per vertex or per pixel level. That gives complete control over the look of surfaces.

Some effects like reflection and refraction just can't be done with a simple texture maps and a fixed-function pipeline.

Programmable shaders are the stuff you put into the GriGri bag to make beautiful real-time 3D voodoo.



Vertex Shaders

vertex shaders can be used to move/animate verts

Linear Interpretation of vertex lighting values

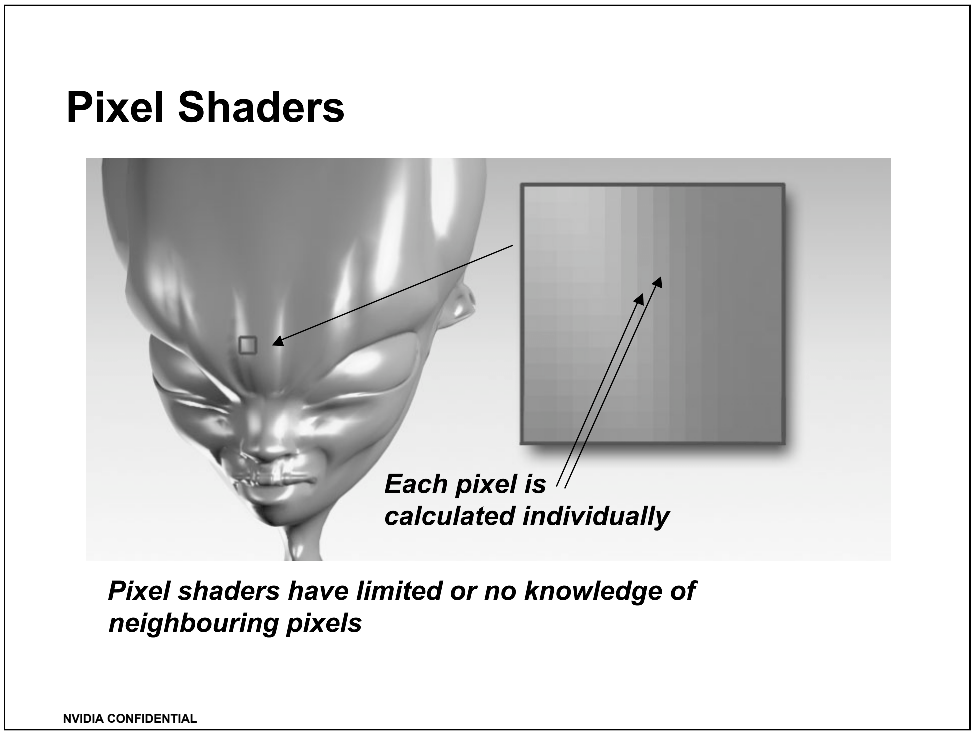
Vertex Shaders are both Flexible and Quick

NVIDIA CONFIDENTIAL

Not always obvious to an artist whether a shader is a vertex shader or a pixel shader.

In general, pixel shaders execute faster since they operate on a vertex level which is far less data-intensive than working on a pixel level.

Vertex shaders execute before pixel shaders and they can both move vertices and shade a surface. For shading, vertex shaders offer less quality than pixel shaders because they can only do linear interpretation between vertex values.



Pixel Shaders

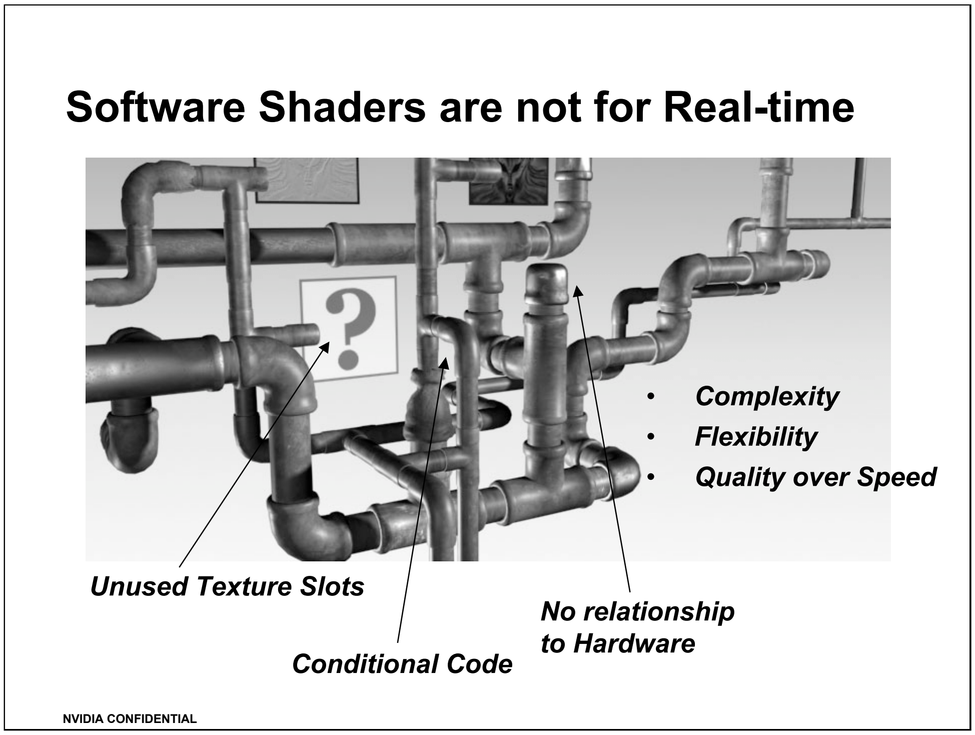
Each pixel is calculated individually

Pixel shaders have limited or no knowledge of neighbouring pixels

NVIDIA CONFIDENTIAL

Pixel shaders calculate the color for any given pixel. Generally the final pixel color is a function of the shading equation and the location of the camera, light, and pixel location in screen space. Most shaders don’t retrun values to the program. They just affect the final screen color. For some complex effects Multipass shaders may be used. Multipass shaders require CPU code to manage the effect.

Pixel shaders affect both diffuse components of the surface as well as the specular components. Of course, pixel shaders can also create any number of lighting models realistic and non-realistic.



Software Shaders are not for Real-time

- *Complexity*
- *Flexibility*
- *Quality over Speed*

Unused Texture Slots

Conditional Code

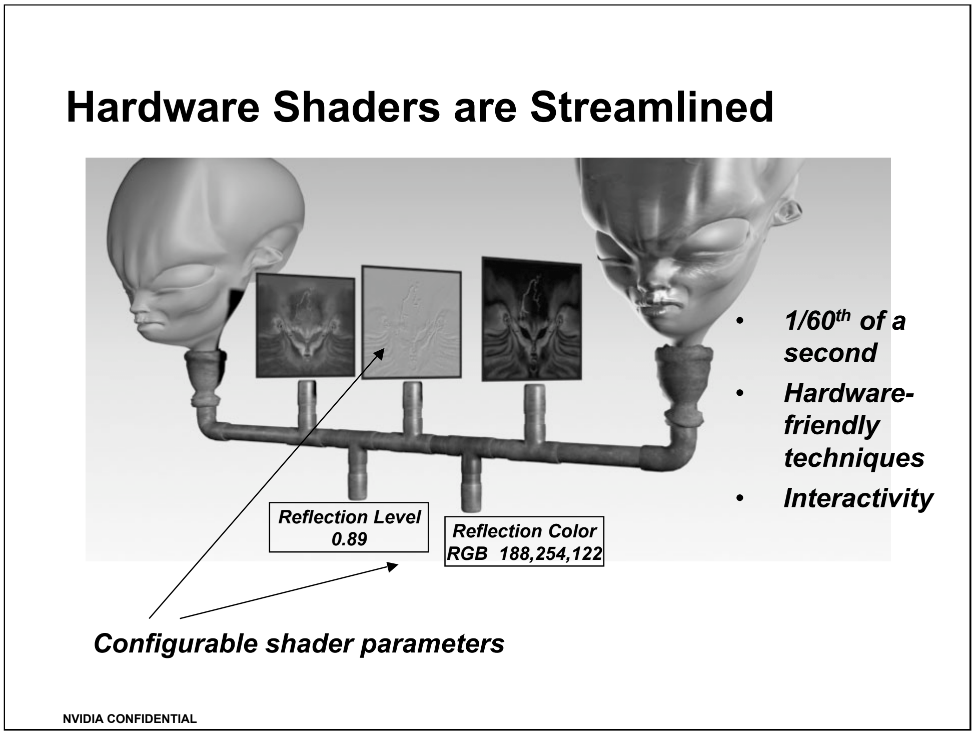
No relationship to Hardware

Artist visualization. Not necessarily scientific.

You cannot expect a shader with an arbitrary level of complexity and flexibility to run in real-time

Software shaders are most often run with just a few maps and features enabled. (You can’t afford this type of inefficiency in real-time)

Rendering speed takes a back seat to the flexibility of a shader



Hardware Shaders are Streamlined

- *1/60th of a second*
- *Hardware-friendly techniques*
- *Interactivity*

Reflection Level 0.89

Reflection Color RGB 188,254,122

Configurable shader parameters

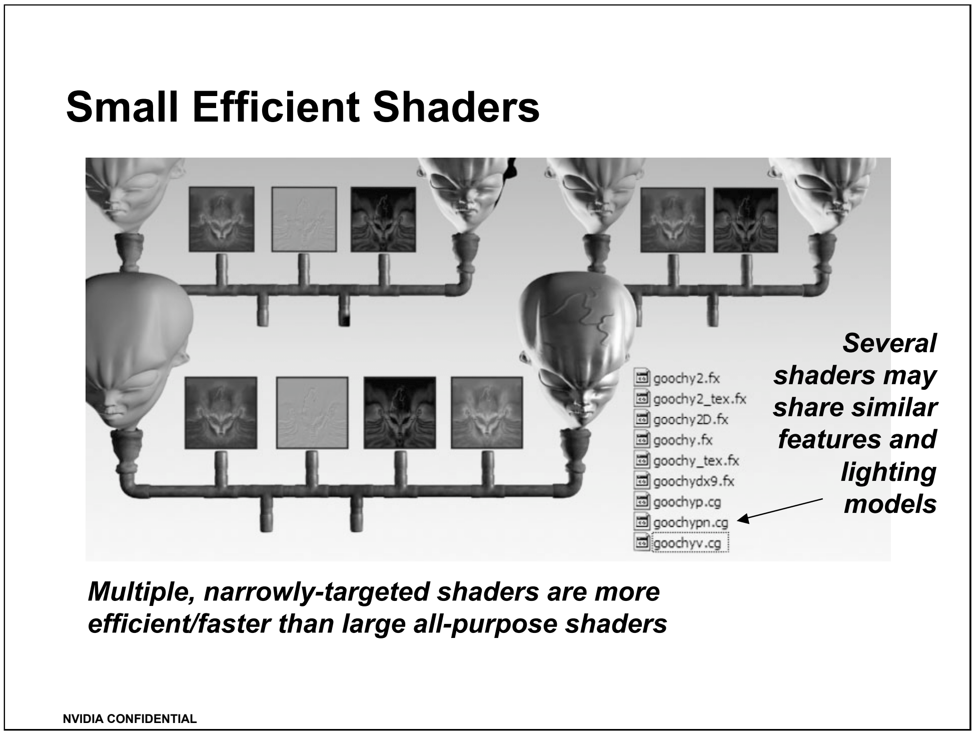
NVIDIA CONFIDENTIAL

Streamlined does not mean simplistic but rather efficient

Variability with a fixed number of parameters. You adjust hardware shaders by specifying different maps and paramter values.

You can’t efficiently disable parts of shader you aren’t using. Best to use different shaders.

Basic plumbing of the shader is determined by the graphics programmer on a project. He/she determines which parts of the shader are artist configurable



Small Efficient Shaders

Several shaders may share similar features and lighting models

Multiple, narrowly-targeted shaders are more efficient/faster than large all-purpose shaders

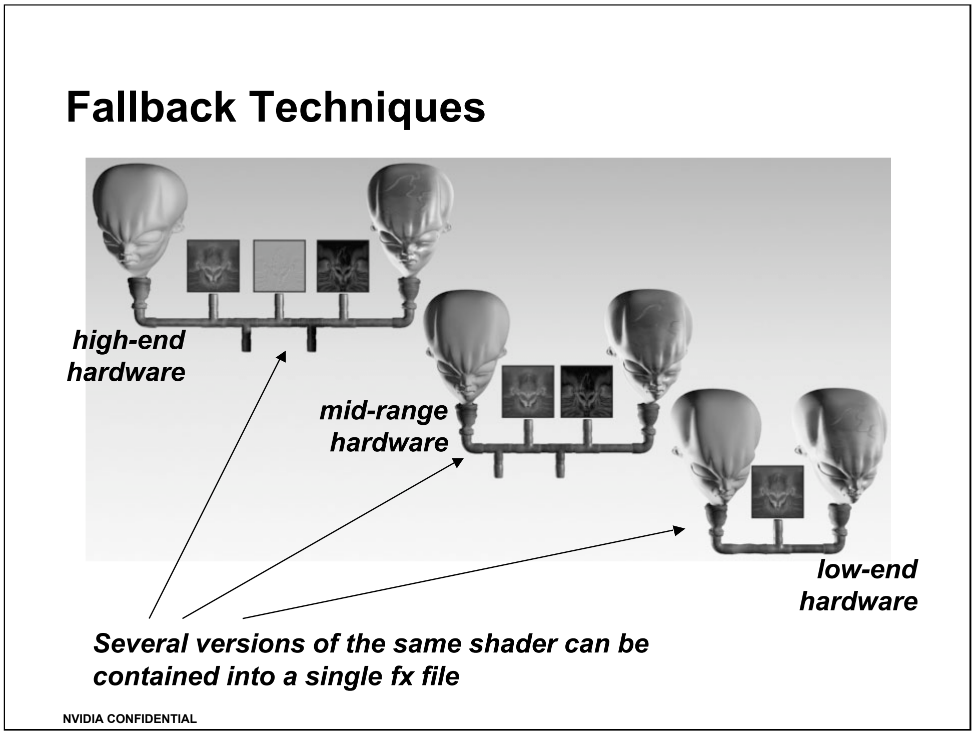
NVIDIA CONFIDENTIAL

Two approaches to writing shaders. First is to have a few very configurable shaders used for everything. The other is to have many small, narrowly-focused shaders.

Hardware shaders more naturally fall into the second category since you want to maximize efficiency. You’ll likely have many slightly different shaders to work with.

Pixel shaders are generally math and texture intensive. More complex shaders become increasingly math-intensive. Shader efficiency is FAR more important than geometric efficiency.

Important also to consider that hardware likes data to be sent a certain way; x number of texture and math operations per clock, can’t just throw random data at full speed. need to optimize. An extra texture may slow the shader down considerably.



Fallback Techniques

high-end hardware

mid-range hardware

low-end hardware

Several versions of the same shader can be contained into a single fx file

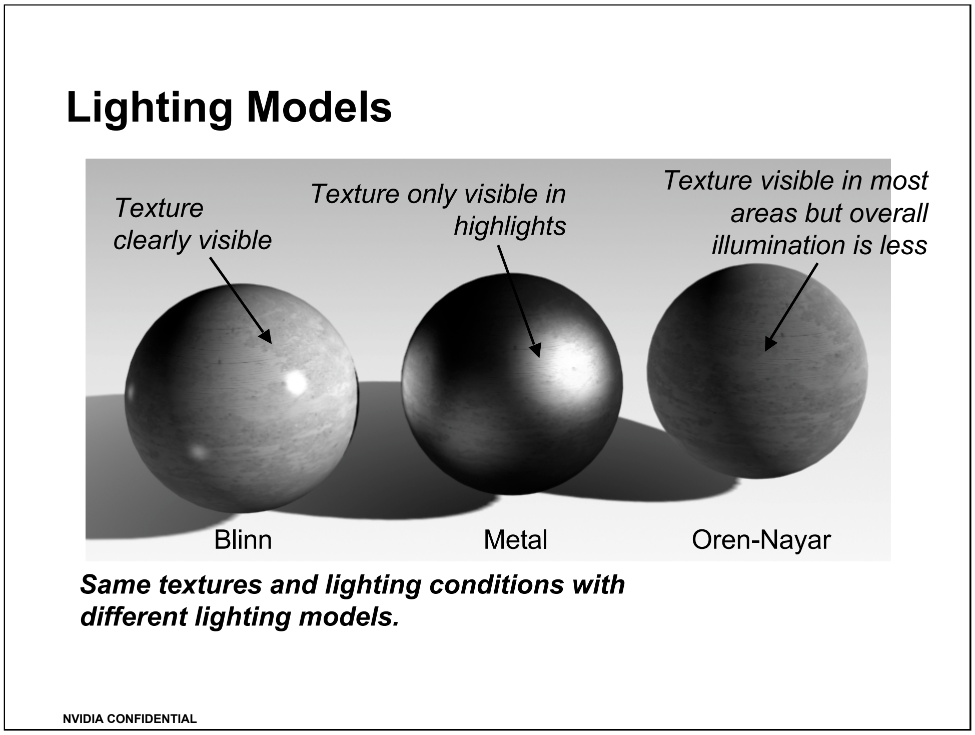
NVIDIA CONFIDENTIAL

Scalability accross hardware, platforms, and LOD levels

Hardware Shaders must generally support graphics chips with different capabilities

Fallback techniques are shaders targeted for different platforms or different levels of hardware capability

A single shader can contain several or no fallbacks. Techniques can be written in Cg or assembly.



Lighting Models

Texture clearly visible

Texture only visible in highlights

Texture visible in most areas but overall illumination is less

Blinn

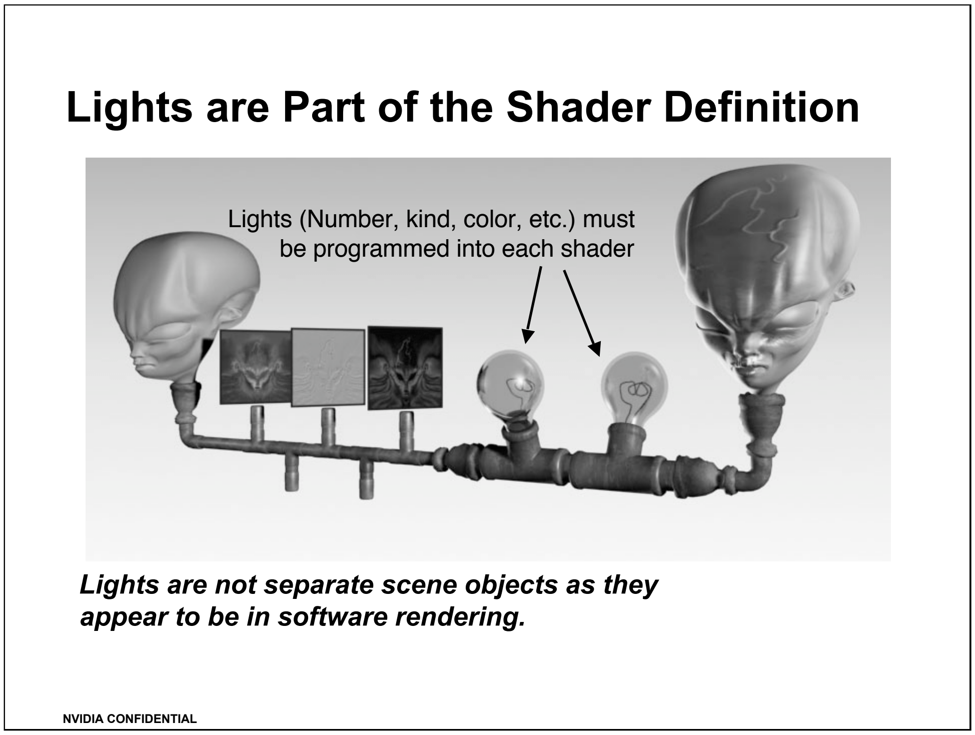
Metal

Oren-Nayar

Same textures and lighting conditions with different lighting models.

NVIDIA CONFIDENTIAL

The lighting model is only part of what makes each shader unique. Multiple lighting models can be combined in a given shader. Most lighting models are a combination of normal angle, eye angle and light direction. Most lighting models attempt to simulate realism with a limited amount of complexity. Non-photo-real lighting models are also common.



Lights are Part of the Shader Definition

Lights (Number, kind, color, etc.) must be programmed into each shader

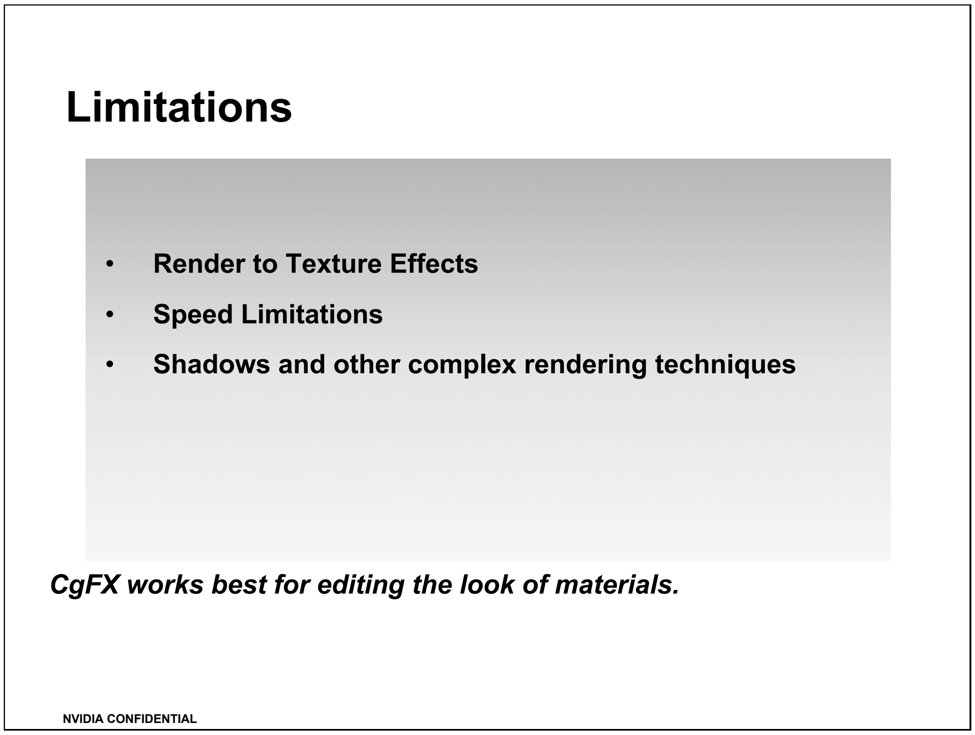
Lights are not separate scene objects as they appear to be in software rendering.

NVIDIA CONFIDENTIAL

Lighting is a function of the shader. The lighting models in use are part of the shader but so are the actual light definitions.

Shaders can light a surface in any arbitrary way; realistic, self-illuminated, etc.

Shaders can be programmed to look for lights in a given scene. What information the shader uses from these lights is entirely dependent upon the shader. Some shaders may take a complete definition. Others may look only at the light position or light color.

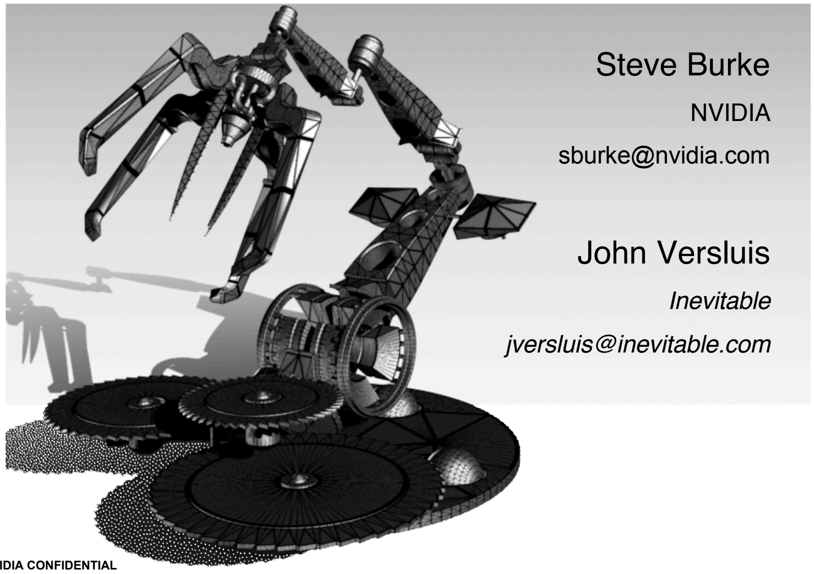


Limitations

- Render to Texture Effects
- Speed Limitations
- Shadows and other complex rendering techniques

CgFX works best for editing the look of materials.

Thanks! Questions?



Steve Burke

NVIDIA

sburke@nvidia.com

John Versluis

Inevitable

jversluis@inevitable.com