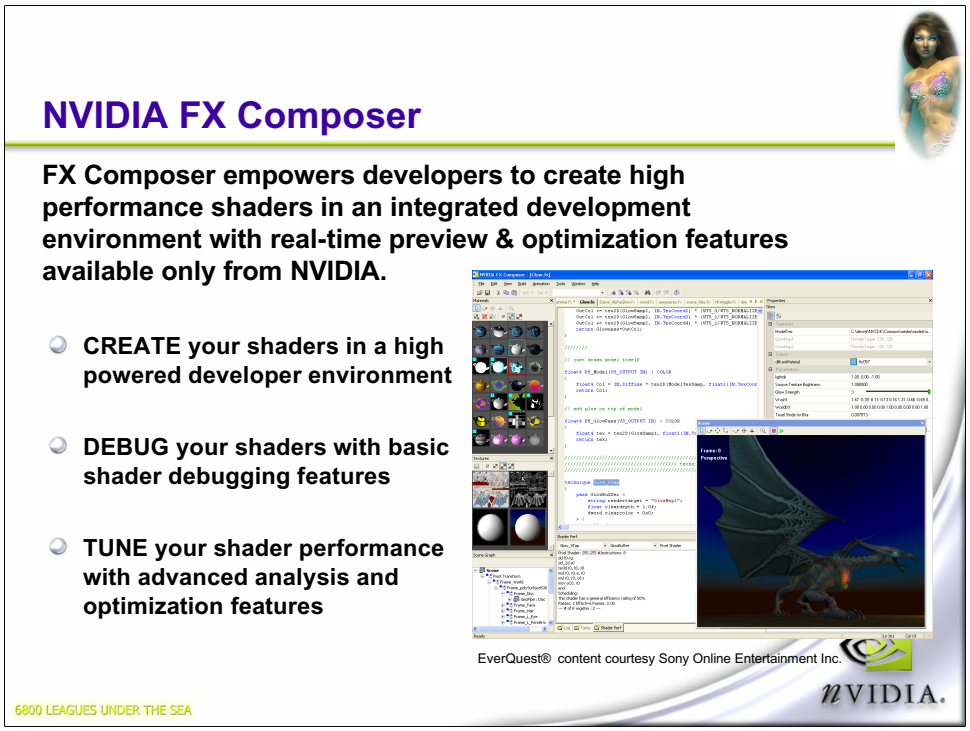
6800 LEAGUES UNDER THE SEA

NVIDIA

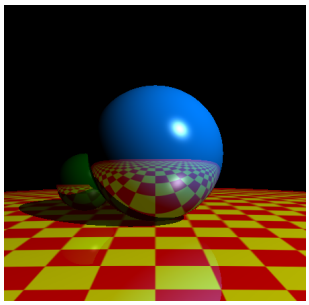
FX Composer 1.5

Chris Maughan



Be sure to check out the complete User Guide and helpful tutorials!

Recap…



- FX Composer – an .fx file IDE
  - Shader development
  - Performance tuning
- FX Composer 1.0 shipped in January
  - ~100 .fx files, ~30 projects out of the box
  - Support for all ps/vs profiles in DX9b
- FX Composer 1.1 shipped around GDC
  - Some new samples, projects
  - A few minor bug fixes
  - http://developer.nvidia.com/fxcomposer

Introducing FX Composer 1.5

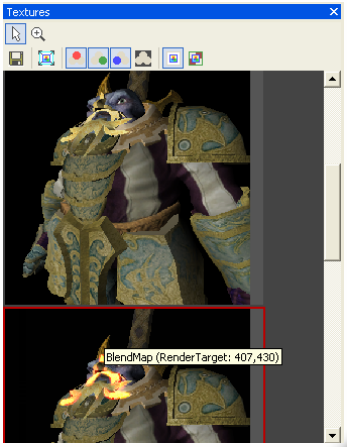
- Lots of new toys…
  - New Application/productivity features
  - DirectX9c support
    - GeForce 6800 & Shader Model 3.0 features….
  - DirectX Standard Annotations and Semantics (DXSAS)
    - Script Execute
  - SDK for plug-ins
  - MSFT .NET scripting through C#, VB.NET

6800 LEAGUES UNDER THE SEA

NVIDIA.

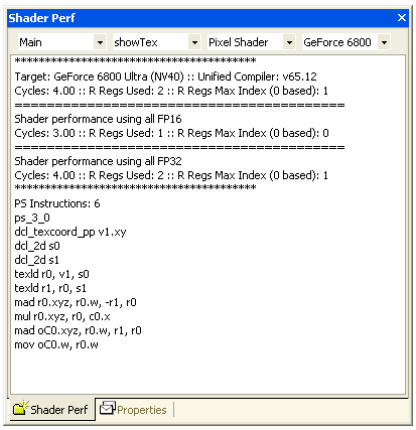
Texture Panel Updates

- Arbitrary Zoom
  - Also on materials panel
- Interactive viewing of scene targets
  - Previously could only see material targets
  - Now can switch to either
  - Real time texture preview
- RGBA color masks
- Better cubemap visuals



NVIDIA.

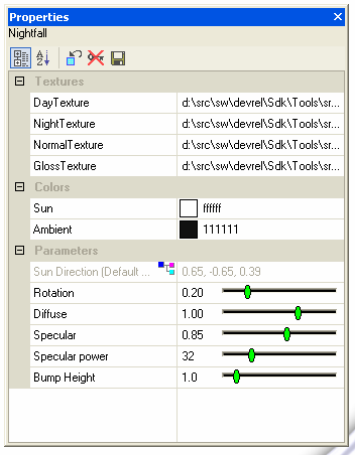
ShaderPerf Panel Updates



- NV4x scheduling
  - Vertex Shader
  - Pixel Shader

Properties Panel Updates

- "Reset to defaults"
  - Reset to the values in the FX file
- "Delete keys"
  - Delete all animation keys in the parameters
- Save properties
  - Save all the values to a simple XML format
  - Works for any object with properties

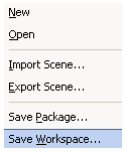


XML Format Example

```
- <connectionparameter objectid="3" name="WorldITXf"
    semantic="WORLDINVERSETRANSPOSE" type="5" rows="4" columns="4"
    handle="134857744" animated="0" numkeys="1" defaulttype="float4x4"
    defaultvalue="0.000000e+000,0.000000e+000,0.000000e+000,0.000000e+000,0.0
  - <keys>
      <key num="0"
        value="1.000000e+000,0.000000e+000,0.000000e+000,0.000000e+000,0.000
    </keys>
    <annotation nametype="9" valuetype="9" name="UIWIDGET" value="NONE" />
  </connectionparameter>
- <connectionparameter objectid="3" name="WvpXf"
    semantic="WORLDVIEWPROJECTION" type="5" rows="4" columns="4"
    handle="134857480" animated="0" numkeys="1" defaulttype="float4x4"
    defaultvalue="0.000000e+000,0.000000e+000,0.000000e+000,0.000000e+000,0.0
  - <keys>
```

New File Formats

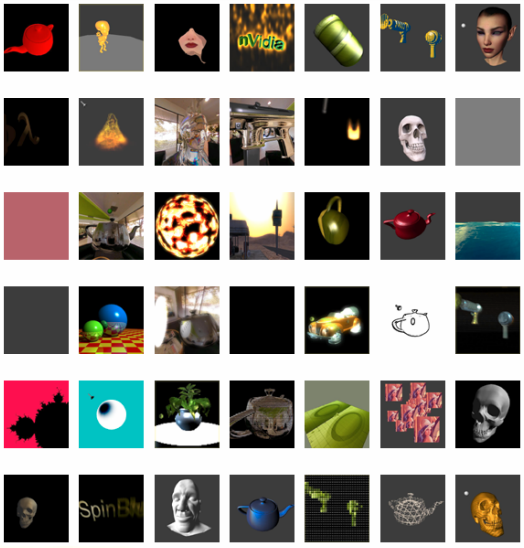
- .fxcomposer files are now known as “Packages”
  - As before, these contain all media, .fx, geometry
- New naked xml file format, known as a “Workspace”
  - See previous slide for example of format
  - Easier to parse
  - Smaller projects, because no media
    - Getting our distribution back below 100Megs…
- .fxcomposer still useful for deployment…
- Also improved package/workspace file management



Improved Project File Management

- Better handling of updated and changed effects that were in the project.
- Better mapping of old to new parameters
  - This is a tricky problem
  - Effect parameters depend on the effect
  - Change the effect, change the parameters…..
- Dialogs to ease confusion
  - Not in the current builds yet

New Projects



- Currently 43 Projects
- … and growing

6800 LEAGUES UNDER THE SEA

NVIDIA.

New examples

- 165 effect files
- … and growing

6800 LEAGUES UNDER THE SEA

NVIDIA.

DirectX9c – NV4x goodness

- FX Composer 1.5 compiled against DX9c
  - Supports HDR format textures
  - DDS floating point also
  - Visible in the textures panel
  - ps_3_0, ps_2_b (and all others)
    - Flow control, etc…
  - vs_3_0 (and all others)
    - Vertex textures
- Many examples shipping with FX Composer 1.5
  - Demo - Sculpt 3D

DirectX Standard Annotations and Semantics (DXSAS)

6800 LEAGUES UNDER THE SEA

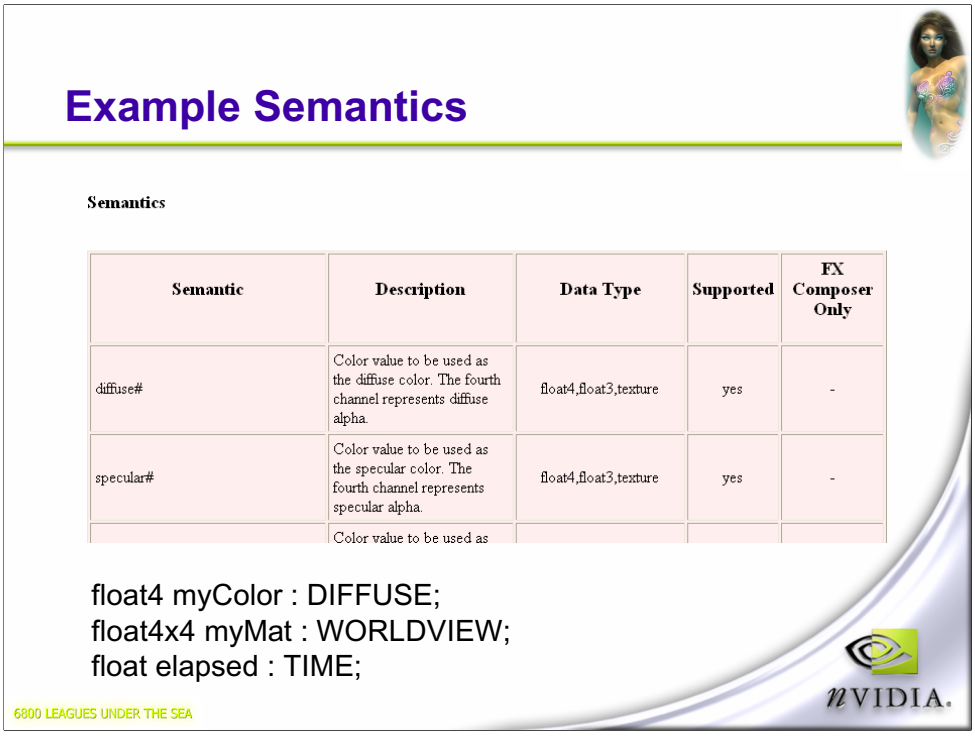
NVIDIA.

Introducing DXSAS

- Specification from Microsoft
- Defines a standard set of semantics and annotations
- FX Composer almost 100% compliant at this point
- Specification will ship around the same time as DX9c
  - So will FX Composer 1.5…
- Help menu brings up the current list
  - You can also use fxmapping.xml to map your own custom annotations/semantics to ours

6800 LEAGUES UNDER THE SEA

NVIDIA.



Screenshot of page from the FX Composer help. This page is generated from fxmapping.xml, which tells FX Composer how to map semantics/annotations to it's internal values, and has validation information for types in fxmapping.

Example semantics – DIFFUSE and WORLDVIEW, conformant with the spec.

Example Annotations

Annotations

Annotation	Description	Data Type	Supported	FX Composer Only
frustum	This is type is associated with a frustum	matrix	yes	-
uiname	This is a string that describes variable, i.e. a pretty name used for labeling an ui dialog	string	yes	-
uihelp	This is the string for helpful information that is displayed to a user in a tool.	string	-	-
uiwidget	Described the widget to be used to edit the value.	string	yes	-

```
float4 myColor : DIFFUSE
<
        UIName="Paint Tint";
        UIWidget = "color";
>;
```

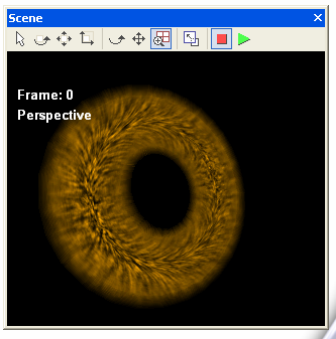

Example annotations.

ScriptExecute

- Major new part of DXSAS
- Designed to solve the effect interaction problem
- Adds powerful scripting features to effects
- A superset of the XML ‘scene commands’ that FX Composer 1.1 shipped with
  - Much more powerful
- All FX Composer effects updated to new format
  - Old scene command XML automatically interpreted as ScriptExecute.

ScriptExecute: Step-By-Step

- **Scripts are added globally, and to techniques & passes.**
- **We'll walk through an .fx designed to be applied to an object first. Fur shells.**
- **All effects start with a global entry point in an effect parameter…**
  - **All compliant effect have ScriptExecute**



6800 LEAGUES UNDER THE SEA

NVIDIA.

Notes for the structure:

STANDARDSGLOBAL semantic to label it. Version number of spec, because it's a float.

ScriptClass is the type of thing that this .fx file handles – in this case it's designed to work on individual objects.

IMPORTANT POINT – the script makes the engine easy because it is always following the script. No scripts without scriptexecute – engine builds a 'fake' one if necessary.

STANDARDSGLOBAL

```
float Script : STANDARDSGLOBAL
<
    string ScriptClass = “object“;
    string ScriptOrder = “postprocess";
    string ScriptOutput = "color";
    string Script = “Technique=Fur;";
> = 0.8;
```


Note: Version number. STANDARDSGLOBAL semantic

ScriptClass – what this effect is designed to work on.

ScriptOrder – Where it is run

ScriptOutput – What it generates.

Script function – just calls a technique script “Fur”. This is like a function call.

Technique Script

```
int shellcount = 12;
int shellnumber;
technique Fur
<
   string Script = “Pass=Setup;"
                   "LoopByCount=shellcount;"
                   "LoopGetIndex=shellnumber;"
                   "Pass=Shell;"
                   "LoopEnd;";
>
{ … } // Pass setup
```

6800 LEAGUES UNDER THE SEA

NVIDIA.

Global variables – shellcount, the number of shells. shellnumber, the current shell we are dealing with.

Same script declaration.

Note the **string concatentation.**

-First, immediately call a script in a pass.

-Now begin a loop, at 12 (shellcount)

-Write the **current loop count into shellnumber**

-Call a pass (Shell)

-End the loop.

-Repeat

Pass Script

```
pass Setup
<
   string script="Draw=Geometry;";
>
{ … }
pass Shell
<
   string script="Draw=Geometry;";
>
{ … }
```

6800 LEAGUES UNDER THE SEA

NVIDIA.

These pass scripts are simple **– just call Draw**.

Draw draws all the **current geometry for this shape**.

Can also call **Draw="buffer"** for a full screen quad.

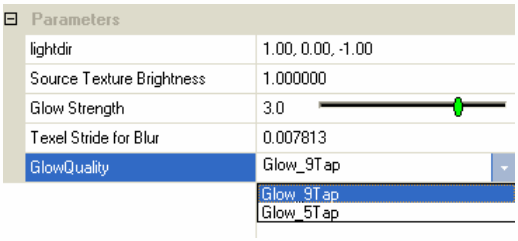
Can also call **Draw="scene"** for whole scene

Time for a demo

Conditionals

- This one from a STANDARDSGLOBAL
- “Glow Quality” appears in UI with a combobox

string Script = "Technique=Glow Quality?Glow_9Tap:Glow_5Tap;";



6800 LEAGUES UNDER THE SEA

NVIDIA.

UI is built from conditionals so the user can choose the path through the script.

Branch Trick

```
bool bReset : FXCOMPOSER_RESETPULSE
<
    string UIName="Clear Canvas";
>;
Script = "LoopByCount=bReset;“ // Run loop?
                "ClearSetColor=PaintClearColor;"
                "Clear=Color0;"
                “RenderColorTarget0=BufMap;"
                "Clear=Color0;"
            "LoopEnd=;"
```

6800 LEAGUES UNDER THE SEA

NVIDIA.

From the paint 3D sample.

FXCOMPSER_RESETPULSE is a private semantic to FX Composer that forces the bool to true at startup, resize.

Otherwise the bool controls the flow.

Declaring targets

```
texture FinalBlurMap : RENDERCOLORTARGET
<
   float2 ViewportRatio = { 0.25, 0.25 };
   int MIPLEVELS = 1;
   string format = "A8R8G8B8";
>;
texture DepthMap : RENDERDEPTHSTENCILTARGET
<
   float2 ViewportRatio = { 1.0, 1.0 };
   string format = "D24S8";
>;
```

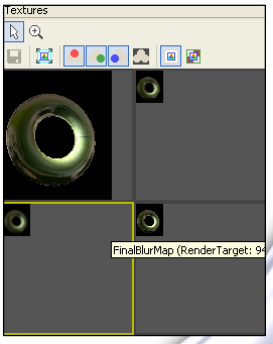

Semantics declare that they are targets.

Ratio used to get relative to viewport

Can also have **Dimensions**.

Using targets

```
float4 ClearColor : DIFFUSE = { 0.0f, 0.0f, 0.0f, 1.0f};
pass GlowV
<
    string Script =
        "RenderColorTarget0=FinalBlurMap;"
        "ClearSetColor=ClearColor;"
        "Clear=color;"
        "Draw=Buffer;";
>
```



NVIDIA.

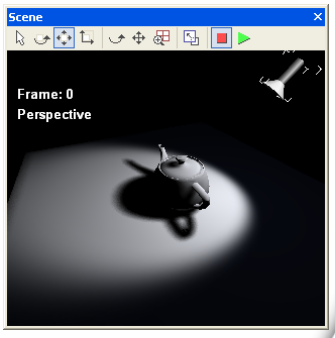
Note – **index** for **rendertarget MRT**

Global Color value – can set in the UI.

Draws a quad.

Renderport

- Rendering from different POV
  - Switch camera from script
- Example scene, Soft Shadows
  - Depth map rendered from POV of light
- Current matrices are changed to use the values from the light



"RenderPort=light0;"

6800 LEAGUES UNDER THE SEA

NVIDIA.

Can do demo at this point, depending on time

LoopByType

- **Enables looping through scene objects**

“LoopByType=Geometry”;
“LoopByType=Light”;
“LoopByType=Camera”;

- **This one not in the Alpha**

Geometry == all objects in scene that use this geometry.

Light == Lights that affect this material.

Camera == Cameras

LoopUpdate

- **Script based update of parameters based on the current loop object specified with LoopByType**

```
float4x4 myMatrix : WORLDVIEW
<
   string frustum="light$";
>
"LoopUpdate=myMatrix@frustum";
```

- **This one not in the Alpha**

6800 LEAGUES UNDER THE SEA

NVIDIA.

Script causes the worldview matrix to be updated with the current light # in the loop.

In beta 1. I'm more hazy about these 2 commands until I've implemented, I won't fully get it.

Hint

Script="Hint=MySpecialEngineCommand";

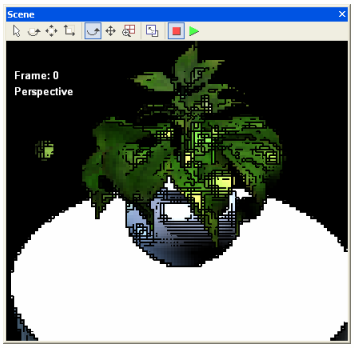
- **Designed to let you add engine specific commands**
- **Beware of using, or you make sharing your effect harder**
- **Useful mechanism though…**

Shader Networks

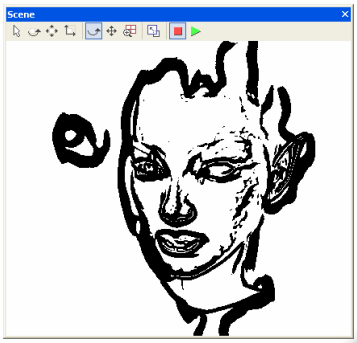
- Effects on their own are cool, but…
- Combining/Interacting effects are the goal
- In the STANDARDSGLOBAL:
  - ScriptClass = “object” or “scene” or “sceneorobject”
  - ScriptOrder = “standard” or “preprocess” or “postprocess”
- Used to determine how this script interacts with other effects
  - We’ve seen “standard” “object” script so far

Shader Stack Demo

FX Composer 1.5 renders a stack of effects



Tiles.fx + EdgeDetect.fx

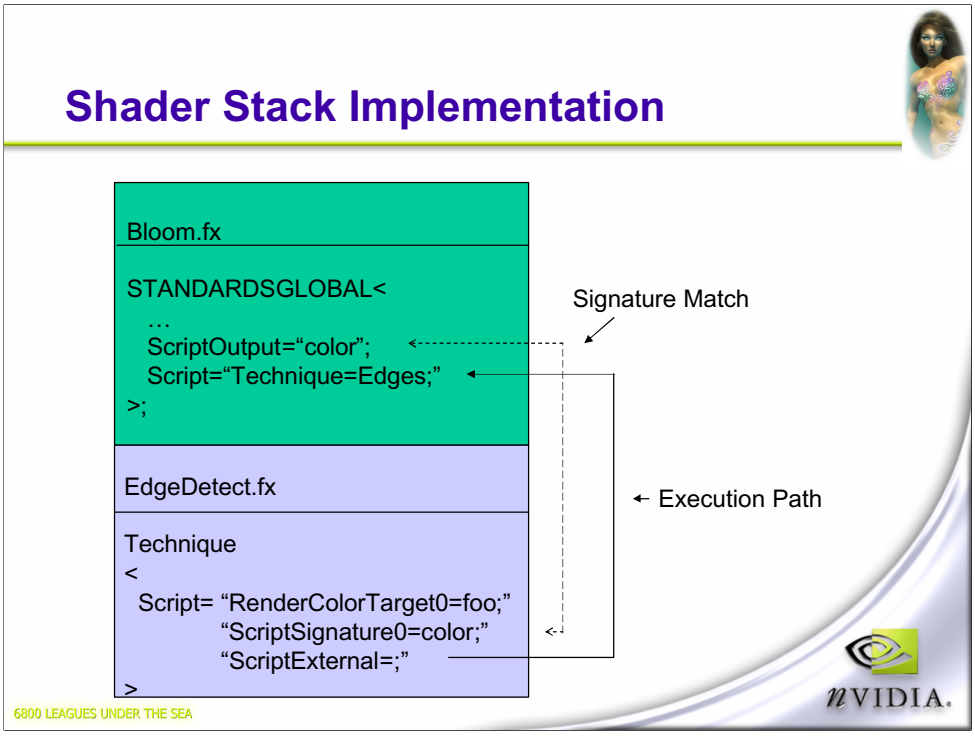


Corona.fx + EdgeDetect.fx

How Does It Work?

- "ScriptSignature" command precedes "ScriptExternal"
  - ScriptSignature = "Color", "Depth", "Normal", "Stencil"
- The signature is used to match up with an external effect
  - Signature values match those of the "ScriptOutput" annotation of an effect
- When the "ScriptExternal" command is called from script, it matches up to an effect with the same signature on it's ScriptOutput value

Diagram coming up



This diagram shows how 2 effects in a stack are matched up based on signature.

The bottom of the stack is ran first, calling the previous stack script to get the color buffer.

ScriptExternal

- The point here is that ScriptExternal jumps to an external effect
- Current Script: “I’m at a point in my script where I need the color data for my effect in this target…”.
  - ScriptSignature0=color, ScriptExternal=;”
- Target Script: “I can provide color”
  - ScriptOutput=“color”;
- Engine enables match up of effects
- Works with MRT (color, normal, etc…)

6800 LEAGUES UNDER THE SEA

NVIDIA.

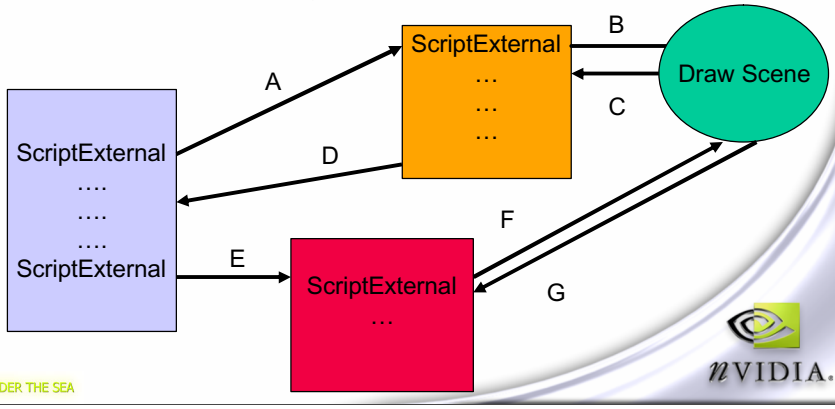
The target color data is placed into the active rendertarget. So a script can declare a rendertarget, then get the current scene into it.

Script Execution Order

- Effects are arranged in scene by artist
- ScriptOrder indicates general nature of effect
  - PreProcess
  - Standard
  - PostProcess
- Engine actually starts rendering at the ‘bottom’.
  - Since the last post-process effect will call “ScriptExternal=“color” and walk up the stack…

Stack vs Network

- **Basic support for ScriptExecute is the stack as described**
- **No reason for a stack, Network is more flexible…**



Follow the arrows to see the execution path. Note that the DrawScene occurs when there's no connection/you're at the top of the stack.

Lends itself very well to wiregraph style of interface…

ScriptExecute Summary

- Very flexible way to solve interacting effect problems
- Not too hard to implement in the engine
- Really powerful
- A Standard
- Will get easier as DCC apps & FX Composer UI catch up…

DXSAS Implementations

- DCC Companies are working on it
  - Varying levels of support likely – some just object scripts, others the Full Monty
- FX Composer will try to implement it all…
  - Already have many effects using it
- Microsoft working on full sample implementation

FX Composer SDK & .NET Scripting

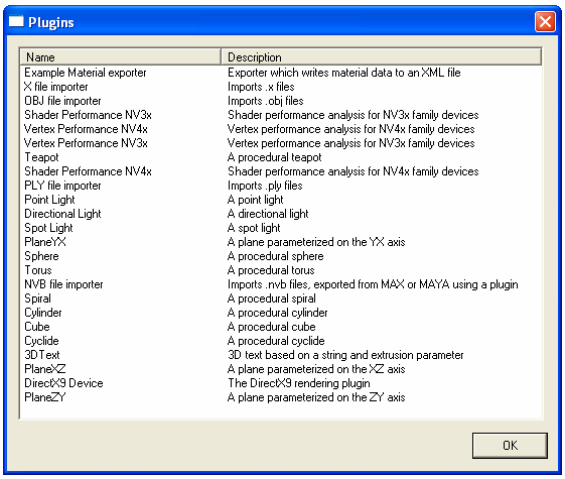
6800 LEAGUES UNDER THE SEA

NVIDIA.

FX Composer SDK

- FX Composer 1.5 is the first version with an SDK
  - Can write plug-ins
  - Can write scripts in C# or VB.NET
- Clean engine, 100% COM interfaces.
  - Easy to build plugins using VC wizard
  - COM nastiness is completely hidden using ATL attributes
    - No code, just a few attributes above the class declaration
  - .NET components can call the engine through RCW
    - Runtime Callable Wrapper
    - Easy cross-communication

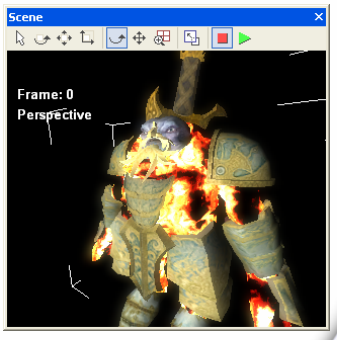
Plug-in Types



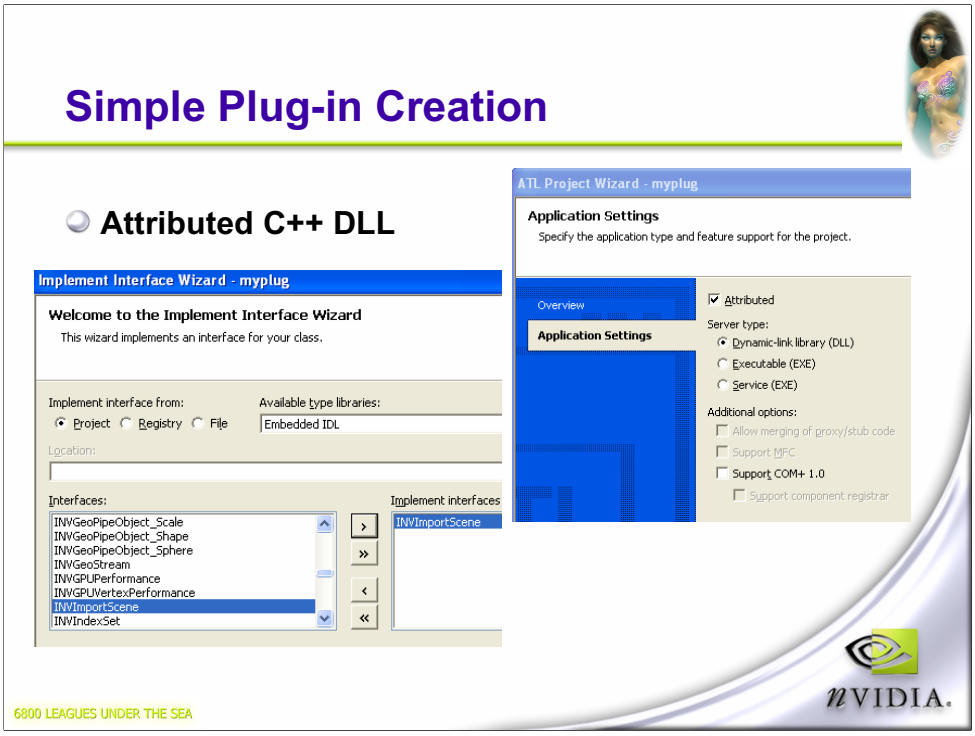
- **Import**
- **Export**
- **Shapes, Lights**
- **Geometry modifiers**
- **Perf analysis**
- **Graphics Device**

Plug-ins For This Release

- For this build import/export a priority
- Most common request
- Example .x file importer, with skinning
- Example material exporter
  - Both samples in nv_pluginexample
  - Full Source



NVIDIA.



You can use the VC IDE to implement the DLL, the COM bits, and let you pick the plugin interface you want to support….

Should we do a ‘custom’ wizard? Not sure this is necessary…

ATL Metadata for an Exporter

```
[
    coclass,
    threading("single"),
    vi_progid("myplug.MyExporter"),
    progid("myplug.MyExporter.1"),
    version(1.0),
    uuid("6A0DBA22-AFE7-448A-8552-66E50EBDA678")
]
Class CMyExporter : public INVSceneExporter
{
    ...
};
```


This is the junk that VC adds to the class definition. All you need to know is that the progid section has to be copied into the FX composer plugins.xml, which tells FX Composer what to look for.

But do you need to write a plug-in?

- .NET scripting is a powerful new feature
  - C# and VB.NET
  - Editing & Compilation integrated into FXComposer
  - Errors displayed in task bar
    - Just like working with an effect
  - Full FX Composer engine is exposed to the script
- Disadvantages
  - No single stepping
  - No intellisense, not currently as integrated into the IDE as 'real' plugins
    - Version 2 fix this

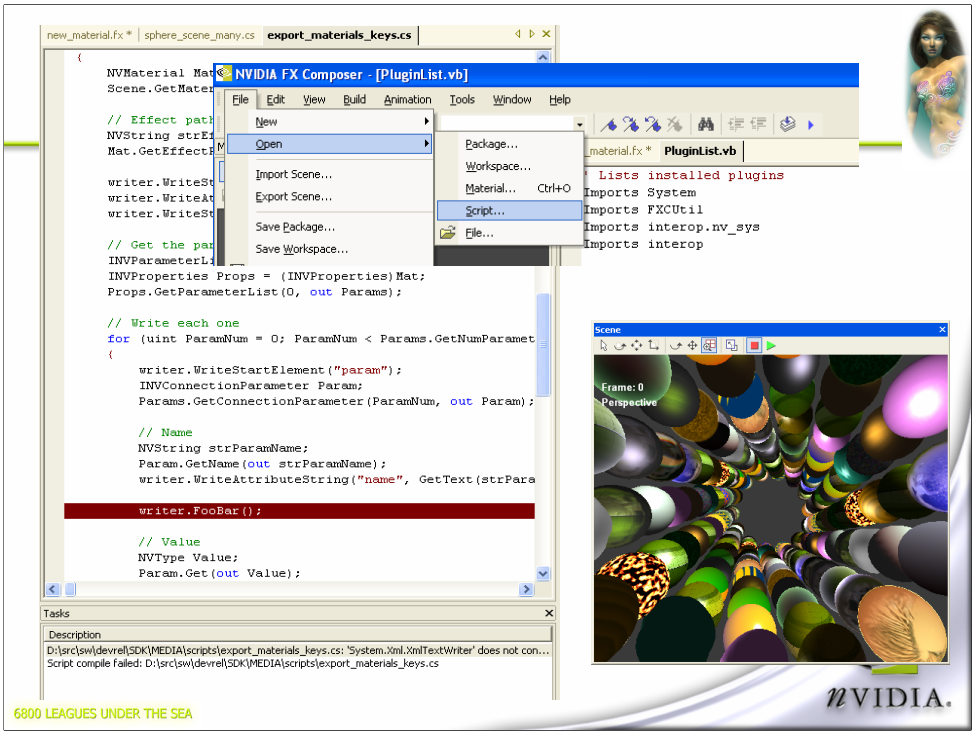
6800 LEAGUES UNDER THE SEA

NVIDIA.

-Debugging of small scripts isn't hard….

-Intellisense is easy to add, but not for 1.5

-May get more integration…. We'll see.



Do a scripting demo, spirals….

Most of the time a script will do...

- Examples
  - Import/export of scene & material data
  - Custom built scenes
  - Material parameter setting/restoring
  - Generation of effect files, based on data
  - Communication between FX Composer & your engine
  - Regression testing, batch processing of materials/effects
    - We have scripts to build screenshots of effects & projects
    - Other samples to copy

6800 LEAGUES UNDER THE SEA

NVIDIA.

If have time, show a script loading projects. Great demo of stability.

Questions?

- Suggestions, bug reports, early access
  - fxcomposer@nvidia.com
  - http://developer.nvidia.com/fxcomposer
- Me
  - cmaughan@nvidia.com
- More FX Composer stuff in
  - GPU Gems
    http://developer.nvidia.com/object/gpu_gems_home.html
  - ShaderX 3

Thanks for listening…

6800 LEAGUES UNDER THE SEA

NVIDIA.