FX Composer 2: A Comprehensive Shader Development Package

Christopher Maughan
Philippe Rollin
FX Composer 2 at a glance

- Rapid Shader Authoring
- Easy Performance Tuning
- Streamlined Pipeline Integration
Rapid Shader Authoring

- Full featured code editor
- Simple Drag&Drop support for most assets
- Artist friendly scene navigation
- Easy shader parameter binding and tweaking
Rapid Shader Authoring
Easy Performance Tuning

- NVIDIA ShaderPerf 2.0 integration
- Easily compare multiple GPU/Driver combination
- User friendly output
Easy Shader Performance Tuning
Streamlined Pipeline Integration

FX Composer 2.0

mental mill AE

CgFX

FX

*.x

*.obj

*.3ds

*.fbx
Mental Mill Artist Edition
Streamlined Pipeline Integration

- Autodesk Maya
- Autodesk 3ds Max
- Softimage|XSI
- mental mill
- FX Composer 2.0
- COLLADA
Production pipeline Integration

- Semantic remapping
- Vertex remapping
- Scripting
- Efficient render to texture
- SDK/Programmability
Semantic Remapper

- Enable remapping of semantic names
- Complex expressions
- Custom semantic conventions for shaders
  - Simply remap names ....
  - ... or create new ones and define how they are calculated
Semantic Remapping Example

\[
\text{SINTIME} = \sin(\text{TIME} + 1.0) \times 0.5
\]
Vertex Remapper

Remapping of geometry input streams

<table>
<thead>
<tr>
<th>Shader Semantic</th>
<th>Type</th>
<th>Stream Semantic</th>
<th>Component</th>
<th>Input Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSITION</td>
<td>Unknown</td>
<td>Custom Remapping</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>POSITION</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NORMAL</td>
<td>NORMAL</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>POSITION</td>
<td>POSITION</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>NORMAL</td>
<td>Unknown</td>
<td>Default Binding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEXCOORD</td>
<td>Unknown</td>
<td>Default Binding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEXCOORD</td>
<td>Unknown</td>
<td>Default Binding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Flexible build configuration

Per project & per effect:
- #defines
- compilation parameters
- Enables ‘uber-shaders’
Scripting

- Python scripting
  - Iron Python .NET

Example Usage

- Rigging lights to materials
- Tweaking material parameters

Scripts are integrated with Undo

```
>>> fxceffect.create("my effect")
<FXComposer.Scene.Effects.FXEffect object at 0x000002A41020>
>>> effect = FxctEffect.OpenEffectFile("bumprefl")
>>> fxcmaterial.setEffect(mat, effect)
>>> for prop in mat.Parameters:
...    print prop.Name
...    print prop.value
Script 0.8
worldMatrix
1 0 0 0 1 0 0 0 0 1 0 -0.612 0.878 0.263 1
wvpMatrix
2.76 -0.549 -0.372 -0.365 4.37e-06 2.41 -0.489 -0.365 0.909 -0.2 0.365 0.144e-06 0.877 0.49 0 -0.416
worldviewMatrix
0.909 -0.2 0.365 0.144e-06 0.877 0.49 0 -0.416
viewInverseMatrix
0.909 1.42e-06 -0.416 0 -0.2 0.877 -0.436 0 0.36
viewMatrix
0.909 -0.2 0.365 0.144e-06 0.877 0.49 0 -0.416
bumpHeight
0.5
normalMap
fxcomposer://localhost/internal#FXSurface2486
cubeMap
fxcomposer://localhost/internal#FXSurface2487
```
Render to texture

- Full Screen Effects
  - SAS .86 – CgFX, FX
  - COLLADA FX
- Can declare shared surfaces
  - Shadow mapping
  - Texture generation
Plugin SDK

Extensible application

Examples:
- Importers - sample source for .x,.obj,.3ds,.fbx
- Exporters
- Remapper examples
- Custom panels/UI
- Render Nodes – modify graph
Schedule

- Public Beta, early April 2007
- Final Release, early May 2007
Conclusion

- Powerful Cross-platform Shader Development Environment
- Streamlined workflow
- Asset Pipeline Integration
Six All-New NVIDIA Developer Tools!
Check it out at our booth (#5134)

- SDK 10
- PerfKit 5
- FX Composer 2
- GPU-Accelerated Texture Tools
- ShaderPerf 2
- Shader Library