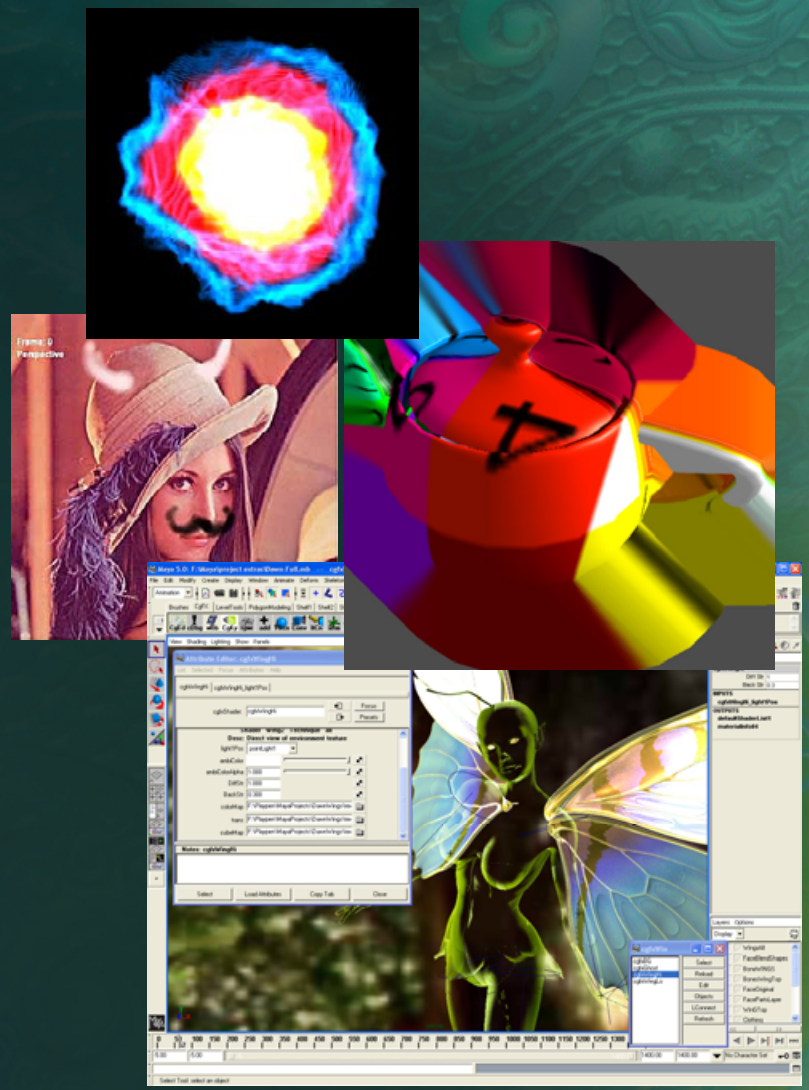
Exposing the SDK

SIGGRAPH 2005

Kevin Bjorke
NVIDIA Developer Resources

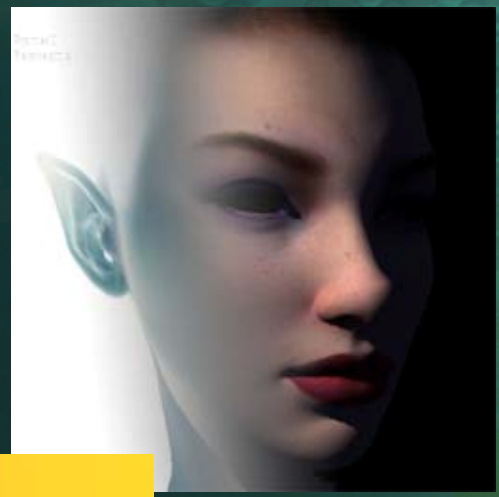
Those “Other” Demos

- NVIDIA SDK & developer site contain:
  - Lots of small demo programs
  - Scores of hardware shaders & effects
  - Assets and plugins for common apps like Maya, Max, XSI, and Photoshop
  - ***SOURCE CODE 4 ALL***



Freely Available

- ~~Get the disk here at Siggraph~~ *SOLD OUT!*
- Grab the latest and greatest: http://developer.nvidia.com/
- Ongoing continuous improvements
- Ongoing new ideas



Color-Space Conversions

Gooch & Cross-Hatch Shadow

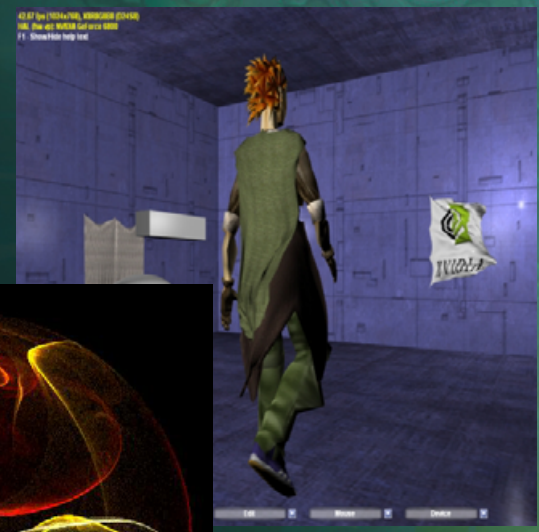
Vertex Repeller

What's in This Talk

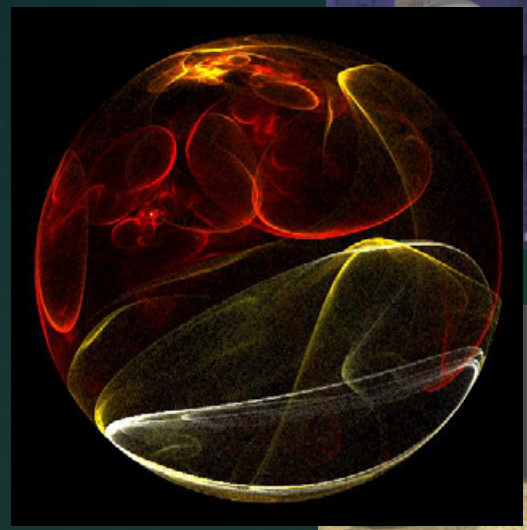
- Some Cool Techniques:
  - G70 Transparency Anti-Aliasing
  - Texture-Driven Fuzz
  - Translucent Shadow Maps
- FX Composer Case Studies
  - "Reice" from Han dae-Hoon: Graphic Factory
  - "Slack!" from Joao Joshua
  - "Spy Girl" from Sami Sorjonen: CGMill
  - "Warrior" from Jukka Tahtinen

Also At Siggraph This Year:

- Be sure to check http://developer.nvidia.com for details on:
  - **GPU Soft Shadows** – Randy Fernando
  - **GPU Cloth Simulation** – Cyril Zeller
  - **GPU Iterated Fractals** – Simon Green



GPU Cloth Simulation



Iterated Fractal

Soft Shadows

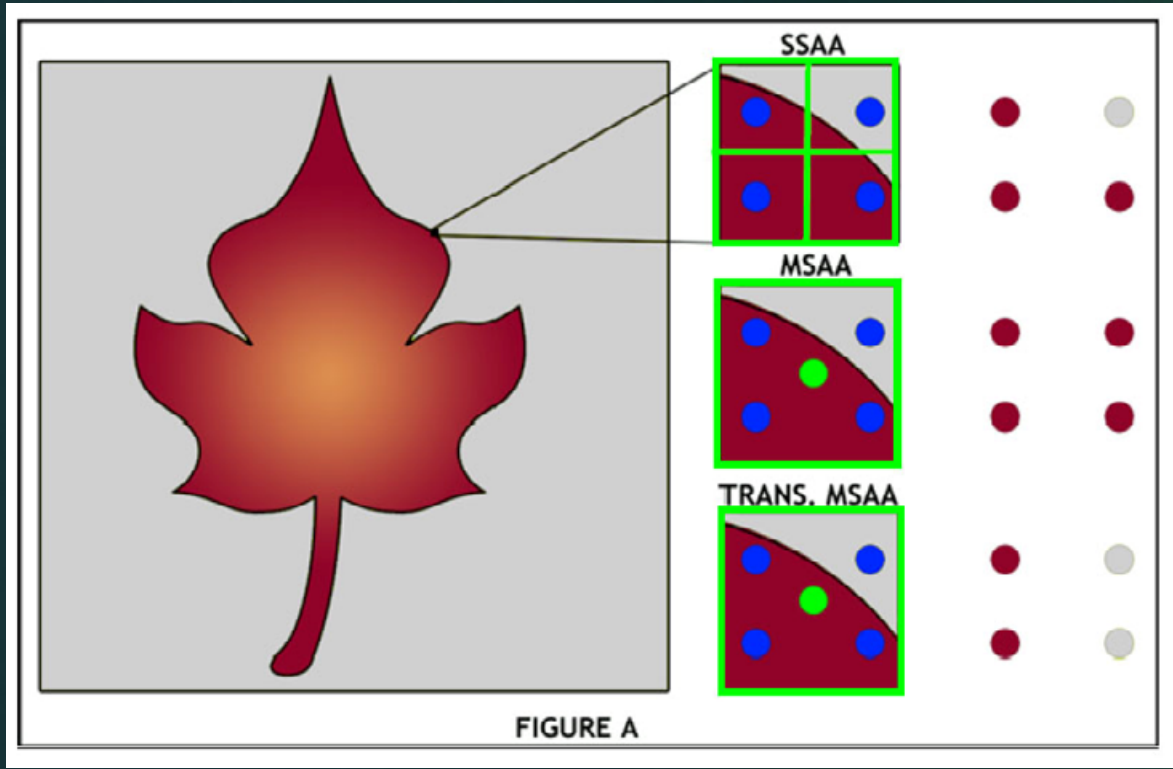
G70: Transparency AA

- IntelliSample™ 4
- Better Quality
- Less Sorting!

"Nature" showing Transparent Anti-Aliased Grass

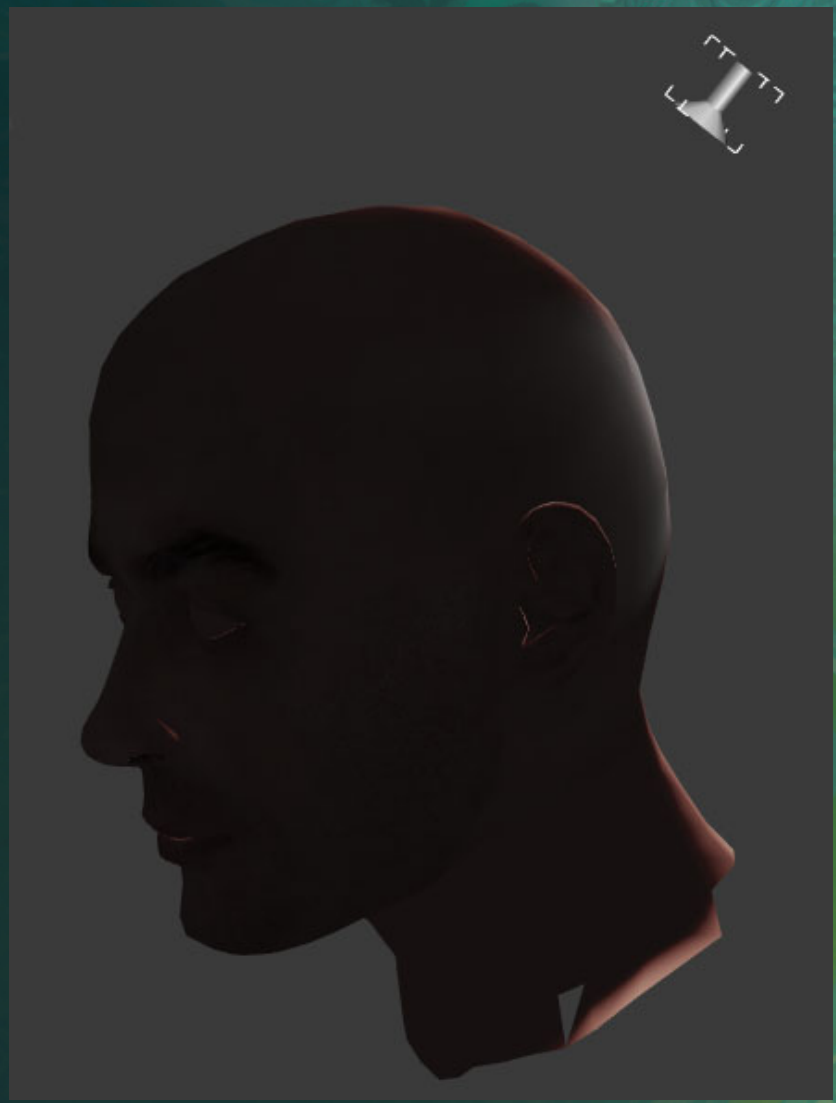
Transparent MSAA

- Factors alpha into the occlusion function for better quality *and* performance



“Asperity” Lighting

DEMO

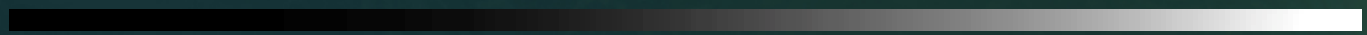


Modeling male pattern baldness as tiny vellous hairs

- Most complex surfaces are covered with small details too tiny to see – esp. hairs, dust, grime, grit
- We can represent the effect of these tiny details statistically
  - cf Goldman et al “Fake Fur Rendering” Siggraph 1997
- These calculations are numerically-intensive
- We can fake it with texture!

Two Parts: Occlusion & Lighting

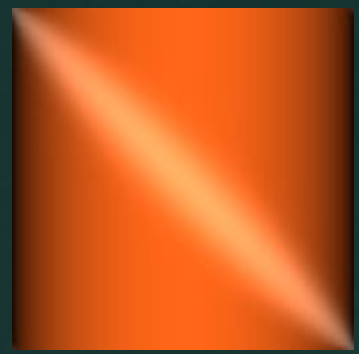
- Occlusion Function
  - At side angles, hairs/dust will occlude the underlying surface
  - Inputs include length/width of occluders, density on surface, and their orientation (“T”)
    - In this simple example of “vertical” hairs we’ll say T=N
    - V = View Direction
    - L = Light Direction
  - Regardless of the numeric inputs, this will be a function of (T·V)
    - We can also attenuate light intensities as a function of (T·L)
  - It has a single numeric domain, so: make it a texture!
    - Represent as a painted gradient
  - We also use the same texture to attentuate light intensities on surface *and* hairs along grazing angles



Simplified Hair Lighting

- The Kajiya-Kay cylinder function models lighting from a transparent cylinderical surface
- It can be reduced to a texture factorized on (T·L) *and* (T·V)
  - T = hair direction vector
  - L = light vector
  - V = v iew vector
- The texture can be generated automatically by HLSL on the CPU, as a grayscale or in full color
- Use “kajiya_kay.fxh” for HLSL
- Set shading parameters via #define before the #include statement

Grayscale

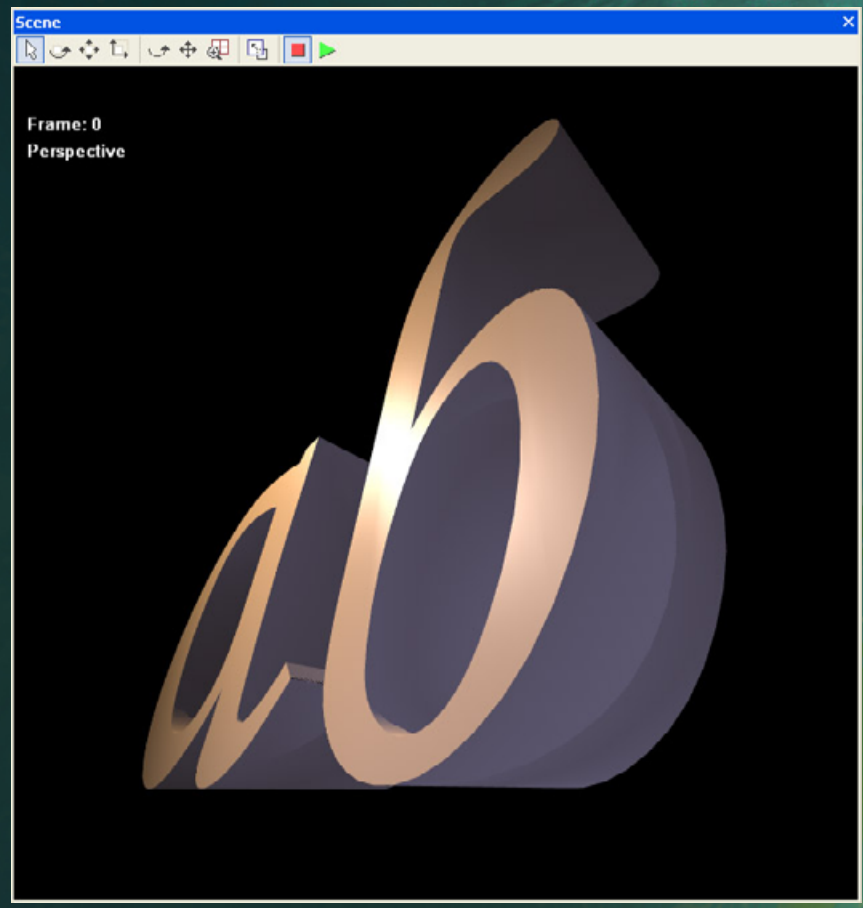


Color

Translucent Shadow Mapping

- Technique similar to that shown by the demo group
- Makes the most of GeForce 6 and GeForce 7 floating-point images
- Demo in FX Composer

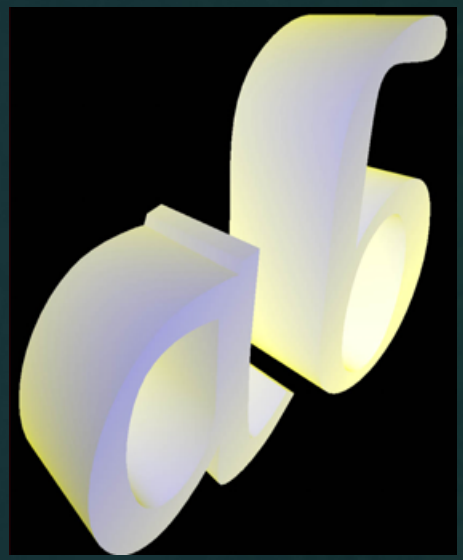
DEMO



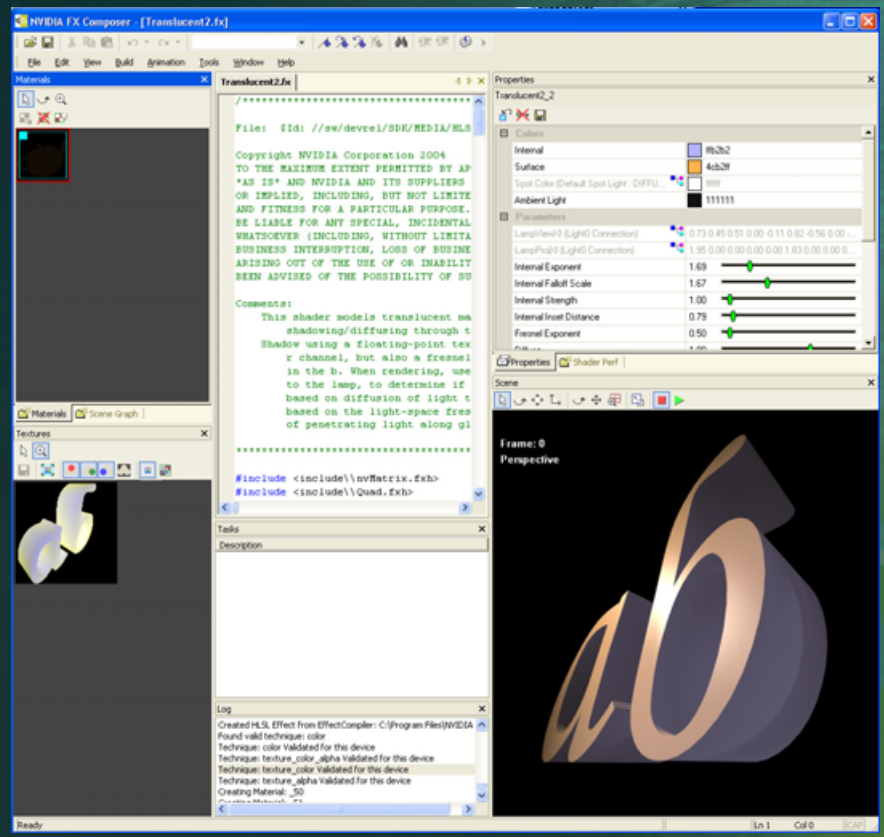
Translucent 3D Letters

Translucent Shadow Mapping

- Each part of the shading pipeline can be seen in this effect



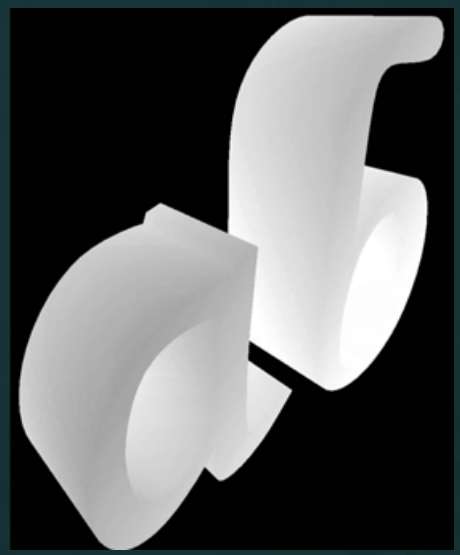
Multi-Channel Shadow Map



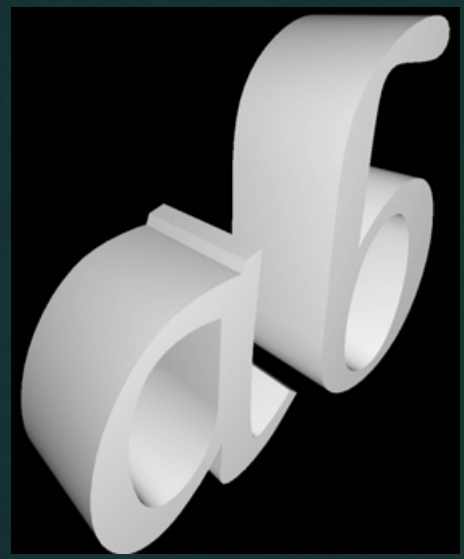
Full FX Composer View

FP shadow map with two parts:

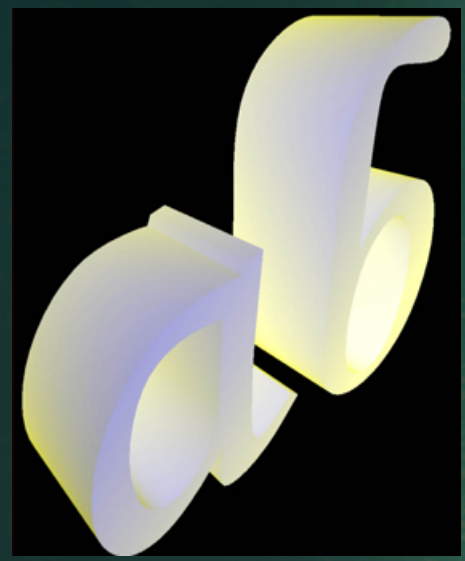
- Red channel contains depth
- Green channel just copies red (for now)
- Blue channel contains fresnel-falloff term



Depth



Fresnel Attenuation



Combined

Moving to Production Models

- FX Composer Case Studies
- Models modified in the past few days based on SDK materials and effects
- Important restrictions:
  - Avoid requiring new art assets
    - Production lead times sometimes restrict the ability to change models and/or textures
  - Make it look better without significant CPU impact
    - Most programs are increasingly “CPU-bound”
    - Using the GPU instead adds richness without CPU penalties

Reice

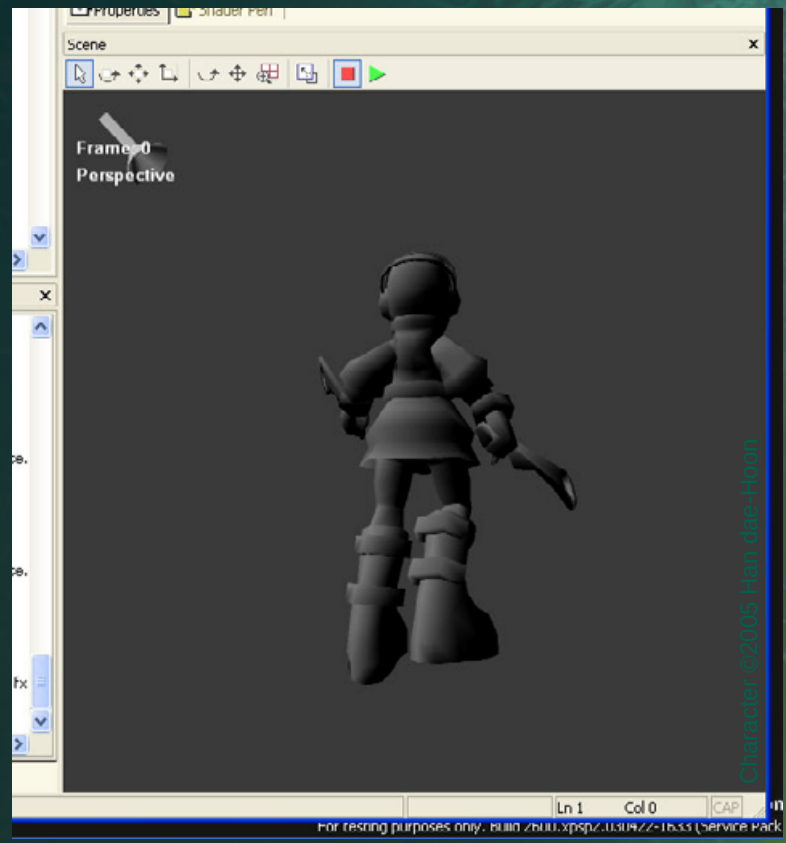
- Made by Han dae-Hoon at Graphic Factory, Korea
- http://www.g-hangun.com/

DEMO

Reice Animated Fire Power-Up

Reice Exported from 3DS Max

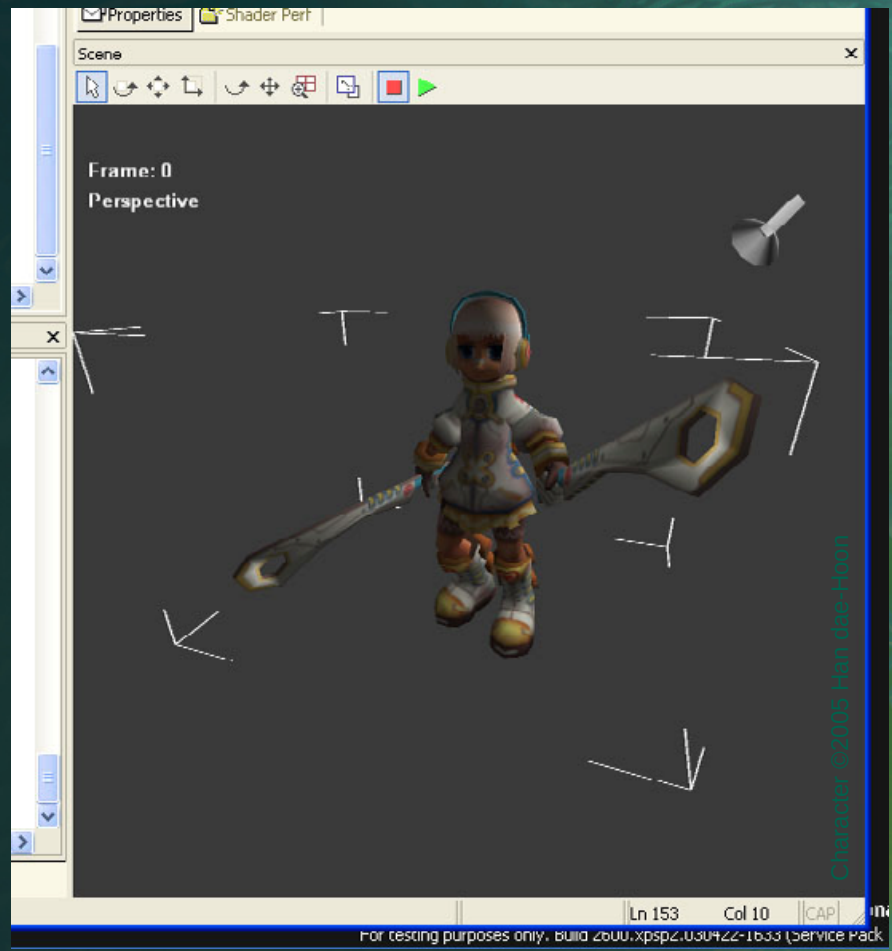
- FX Composer can read .obj/.x/.NVB files
- Export Tips:
  - Group by “Materials”
  - Turn off “Use Relative Indices” in .obj
  - Don’t accidentally export unused/unseen junk like the polys in the 3DS Max “Biped”



Reice as unshaded .obj

Setting Up Base Materials

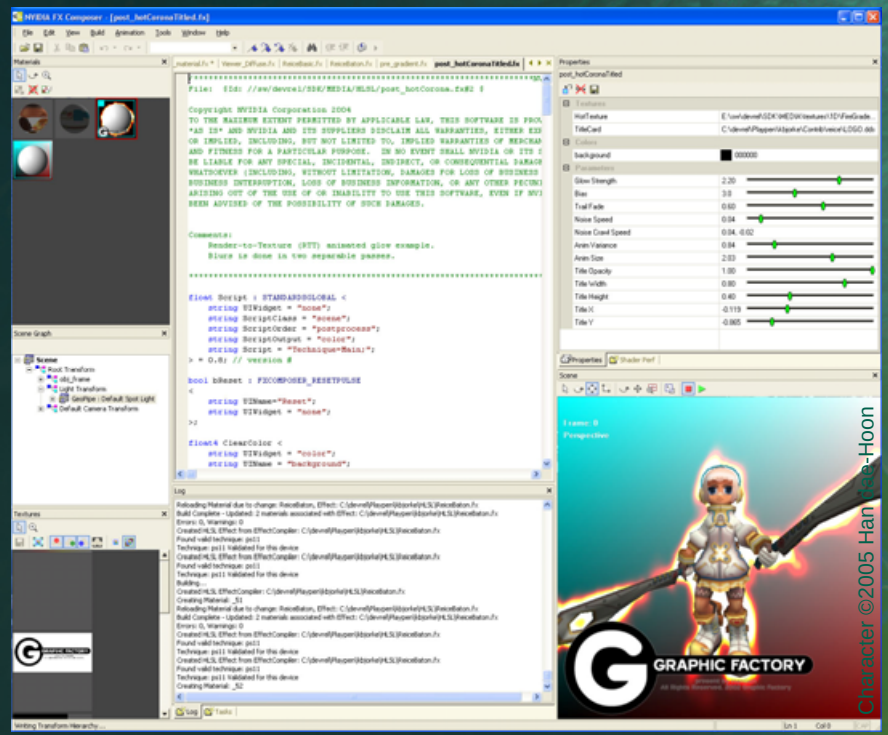
- Trick: material UV coordinates may be flipped in “Y” depending on their origin (OpenGL or DirectX app)
  - So sometimes use “UV.y=(1.0-UV.y)” to flip it
- Two materials here: one uses AlphaTest to modulate hair transparency



Initial Simple Texture Shading

Adding the animated “Power-Up”

- Post-process based on “corona”
  - Grayscale effect with a hand-painted gradient
- Background is a four-point bilinear gradient



FX Composer Display

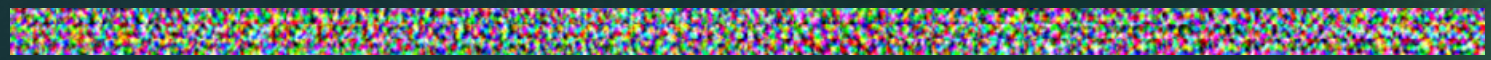
First Step:

- Render Scene to RGBA, and use A *only*…

RGB A

Displace Alpha

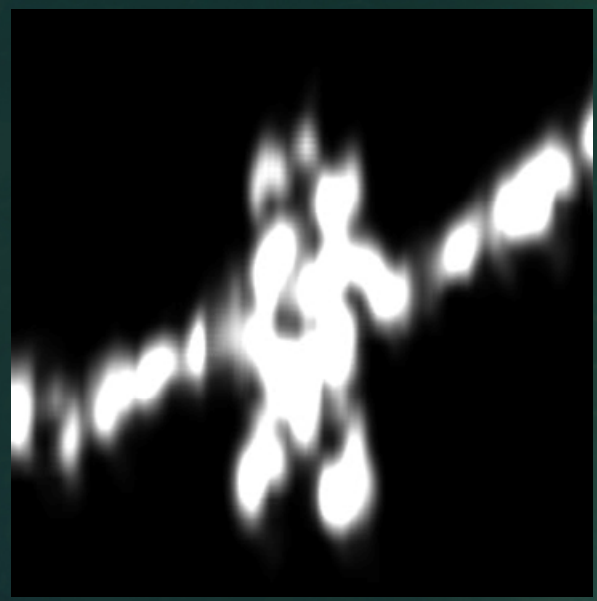
- We lookup signed values in a volume noise texture, and add as offsets to the screenspace-blur U,V values



Flattened 32x32x32 noise

Distorted Alpha + History Buffer

Add to “history buffer” and blur…



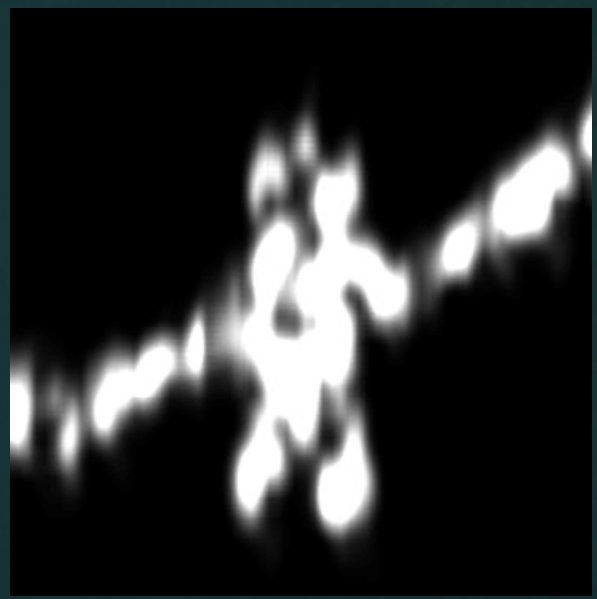
Blurred Grayscale

- We have the previous frame’s blur, so we scale it down (darken it) and add the new displaced alpha to it
- Blur the sum to dissipate/smooth
- Stash-away the result for next frame

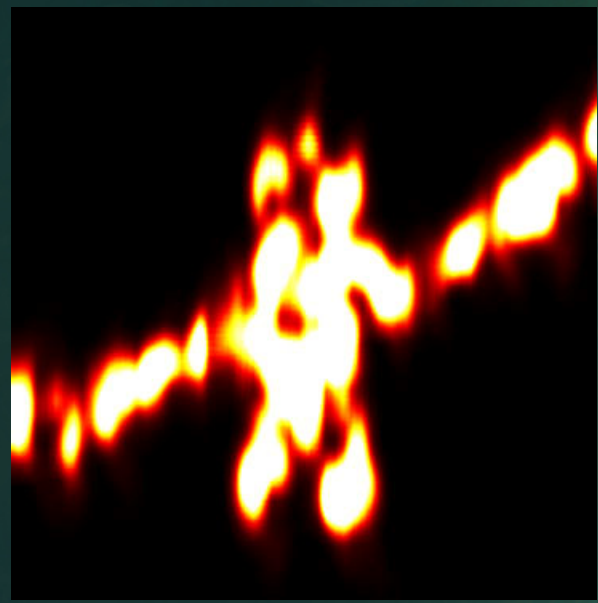
Remap gray 0-255 for color ramp

- Using “black body” color table in Photoshop™ is a good starting point

Gradient (1D Map expanded for visibility)



Grayscale



Remapped to Gradient Colors

Bonus Round: Anime Shadowing

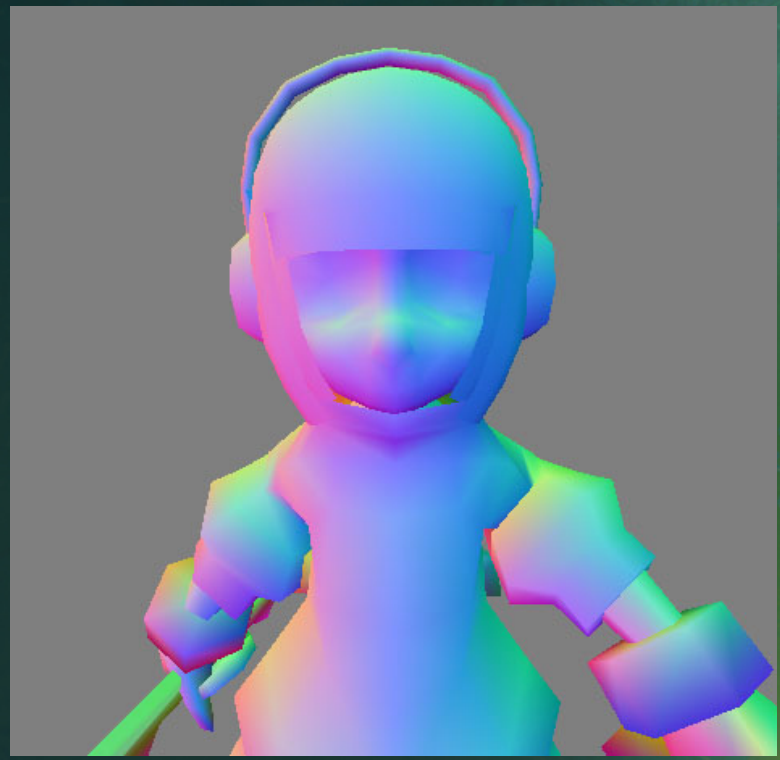
- Inspired by Studio Ghibli's Presentation for SoftImage @ Siggraph 2005
- This is a deferred-shading technique

Reice with Rounded "Hand-Drawn" Shadow Shading

We start with Two RTT passes

- We render color and surface normals w/Alpha

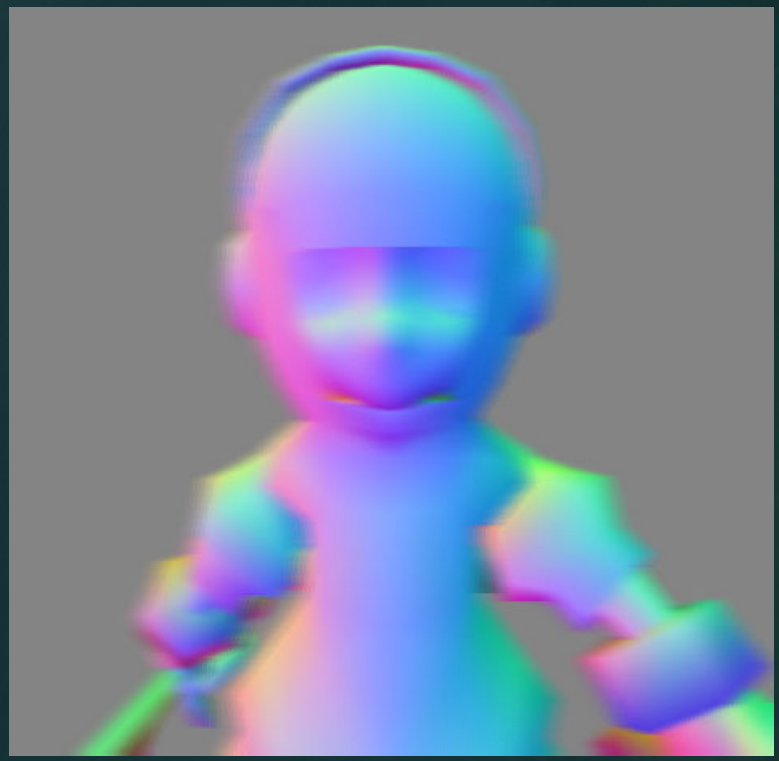
RGB



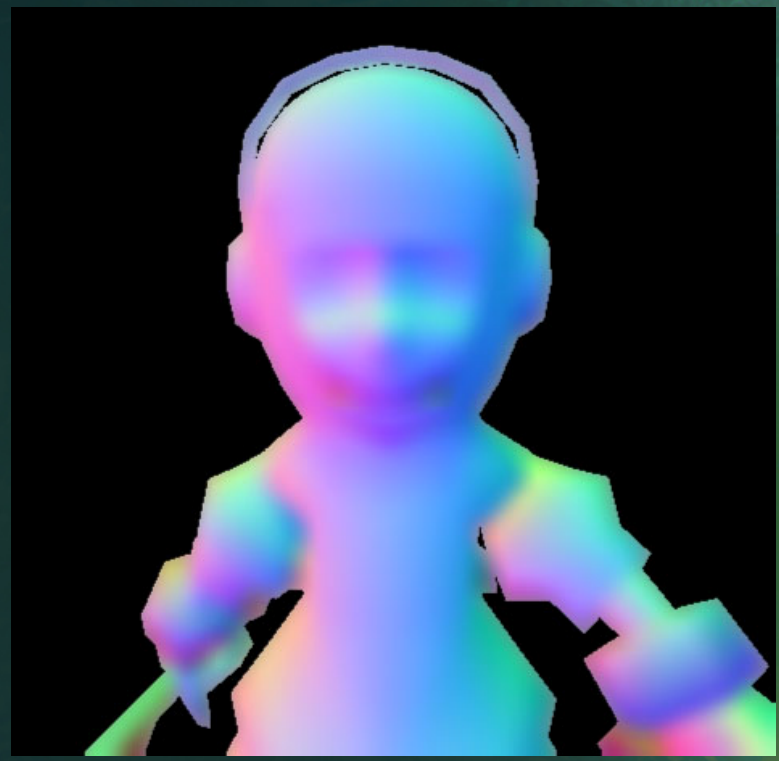
Raw Surface Normals

Next We Blur the Surface Normals

- Optionally, clip against the original alpha

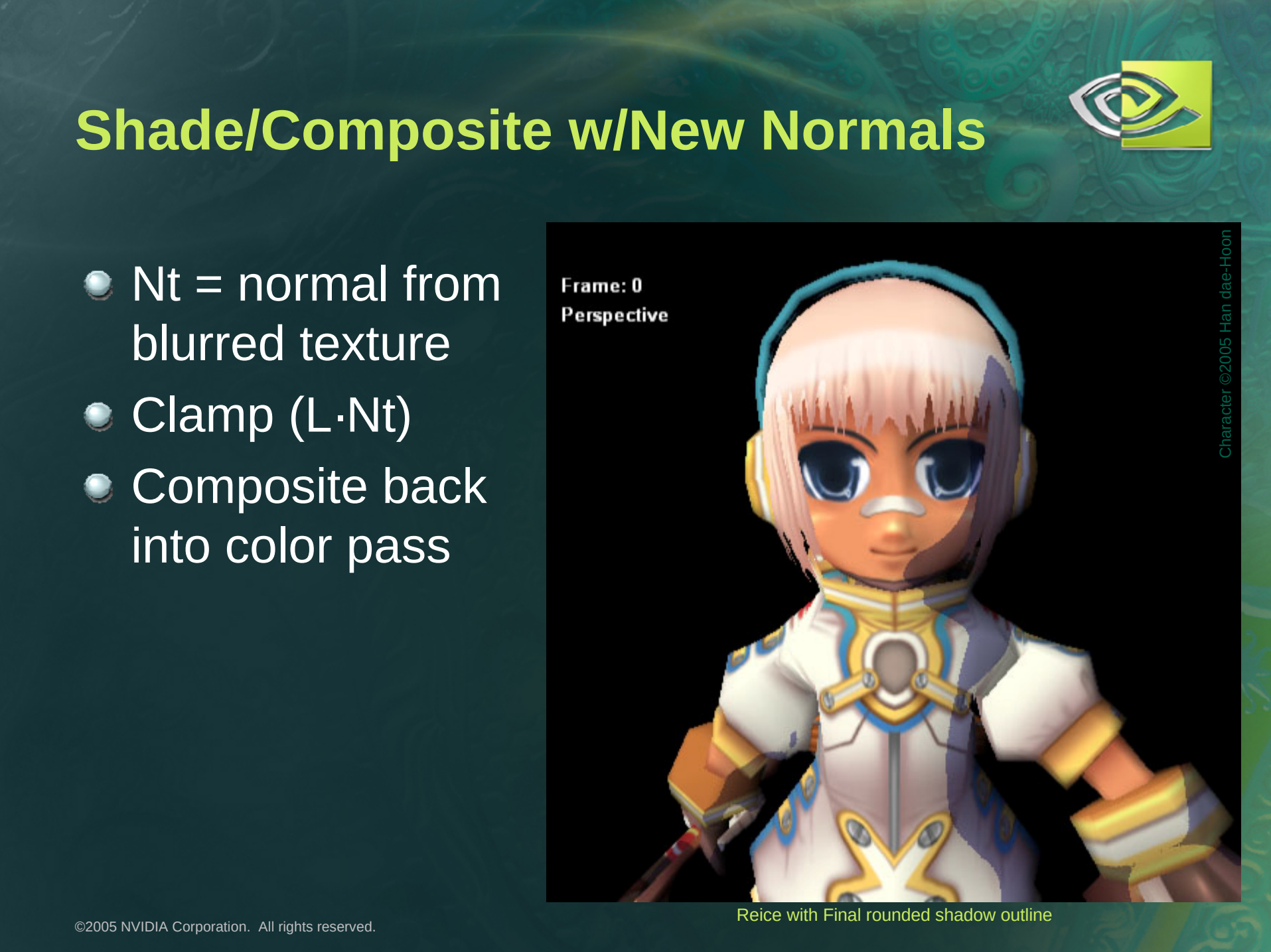


Blur Pass 1: Horizontal



Blur Pass 2: Vertical (and optional clip to show form)

Shade/Composite w/New Normals



- Nt = normal from blurred texture
- Clamp (L·Nt)
- Composite back into color pass

Reice with Final rounded shadow outline

Slack!

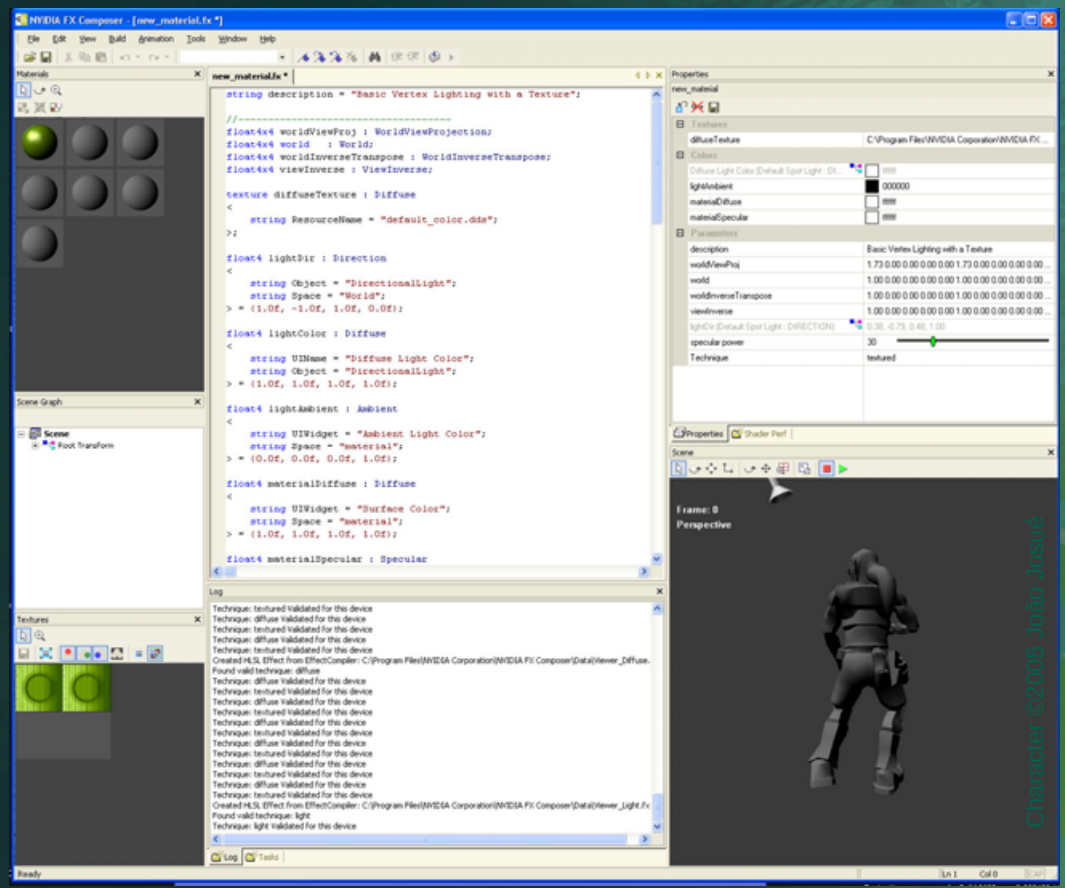
- From artist João Josué of Portugal
- Joao.Joshua@gmail.com

DEMO

Slack!

Default .OBJ Materials

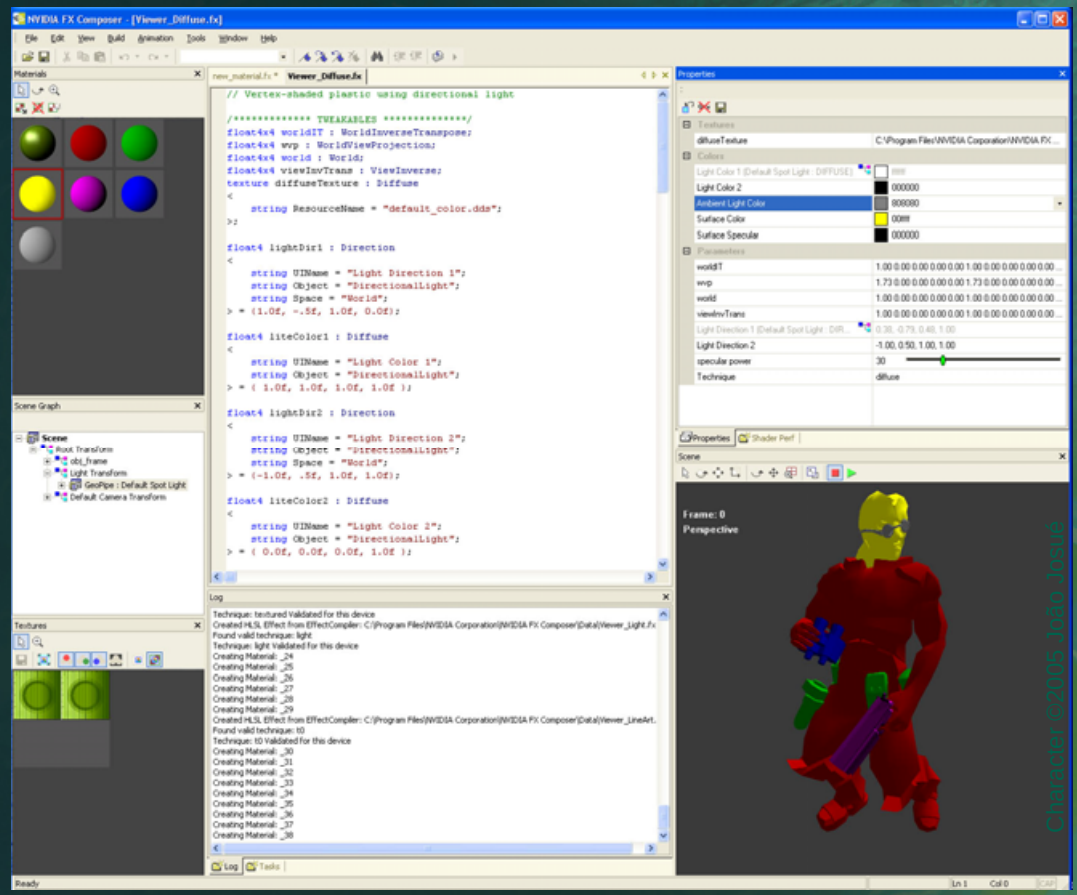
- By default, materials will be gray



Default Scene as imported via .OBJ format – note gray material samples

Making Navigation Easier

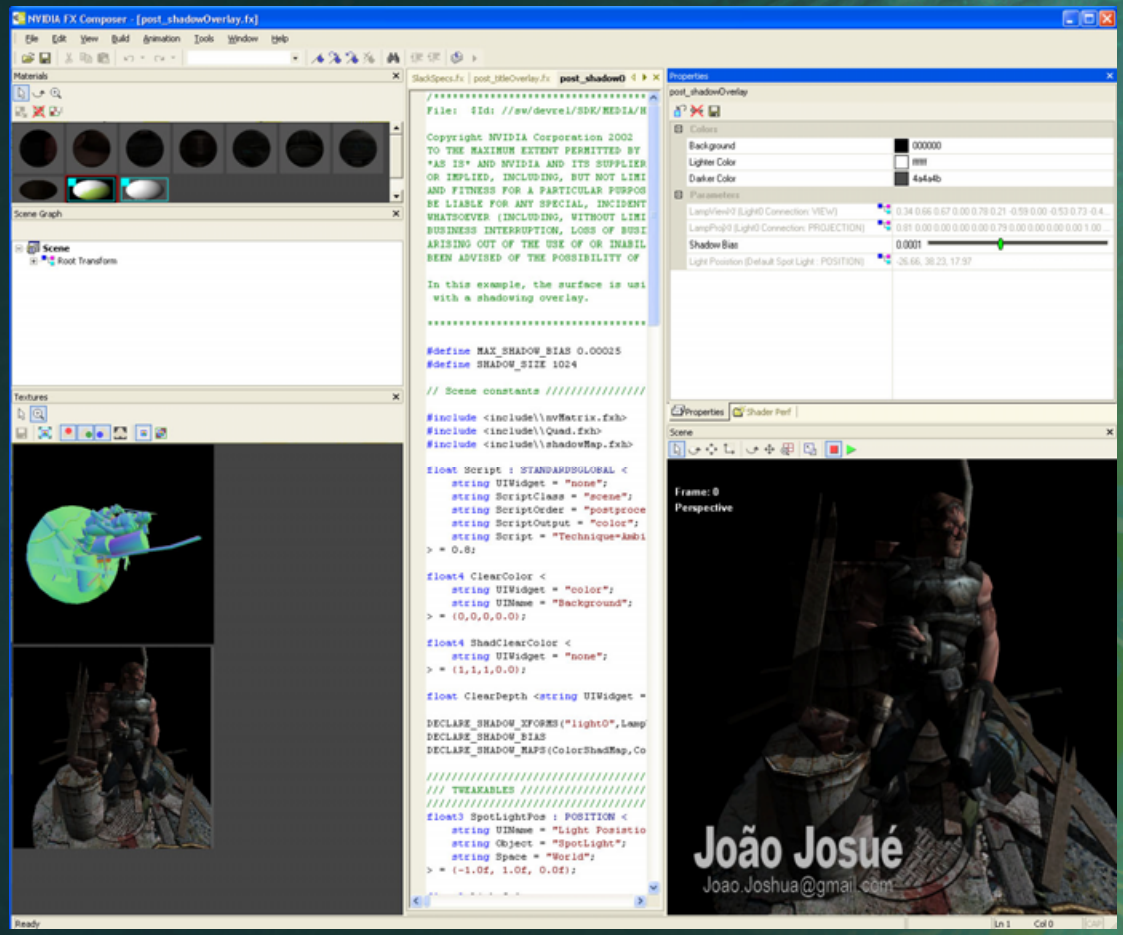
- Setting a color to each will make the model faster to understand and navigate
- We will usually replace all of these:
  - As a group
  - Individually



Components recolored for easy identification and re-assignment

Using an Overlay Shadow

- Materials that create shadow maps can't share maps with other materials
- So we create a single unified shadow map
- Composite as an overlay

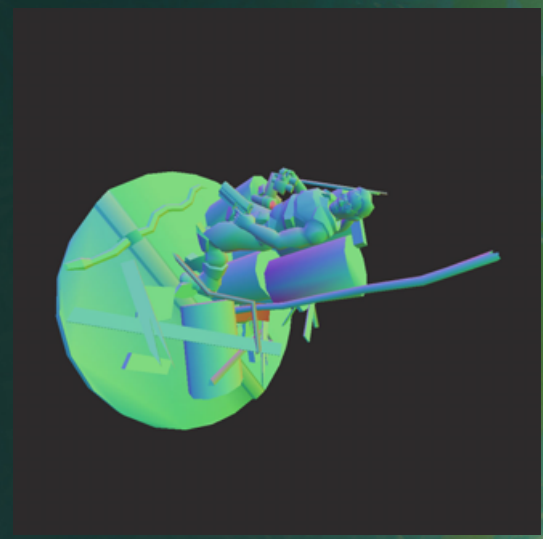


Slack! + Set – All Sharing a Single Overlay Shadow Map

Overlay Shadow Steps (1)

- The unshadowed scene is rendered to an offscreen texture
- All geometry is re-drawn once to create the shadow map
  - using D24S8_SHADOWMAP in this case, but any shadowing method will work
  - Can even be adapted to use stencil shadowing
    - see “softStencilShadows” in the SDK 9.5!

Varying Materials Rendered without Shadows


Shadow-Pass Camera View

Overlay Shadow Steps (2)

- Now re-draw the geometry again
- Use shadow map with usual 3D locations
- Get color from unshadowed render according to screen space

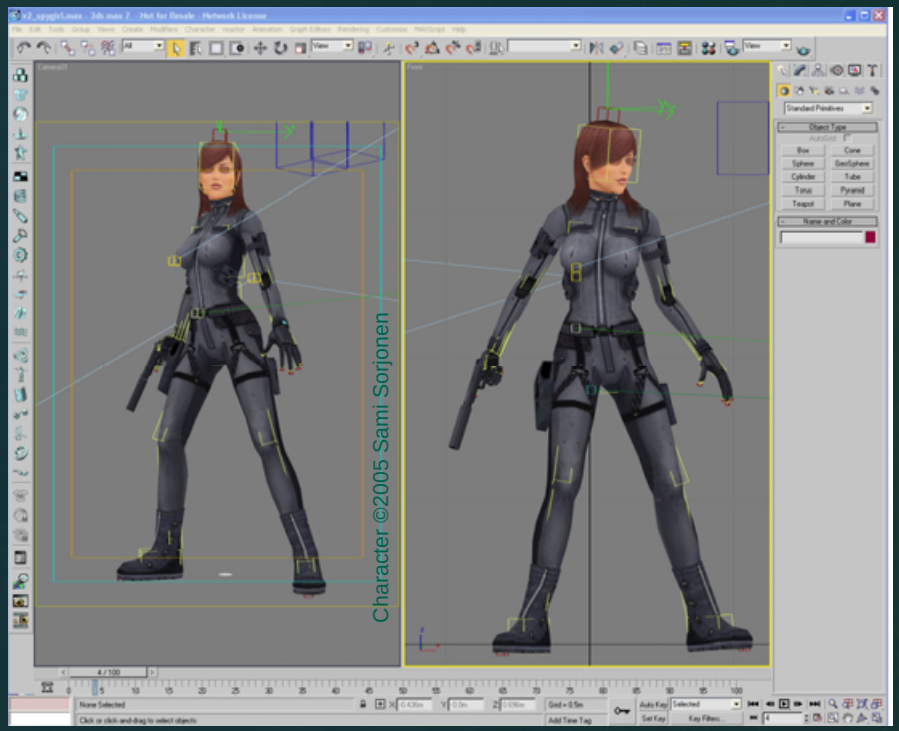
Shadow *Only*

Final Result

SpyGirl

DEMO

- From Sami Sorjonen
- http://www.CGMill.com



Simple Shaded View in 3DS Max

Output from FX Composer

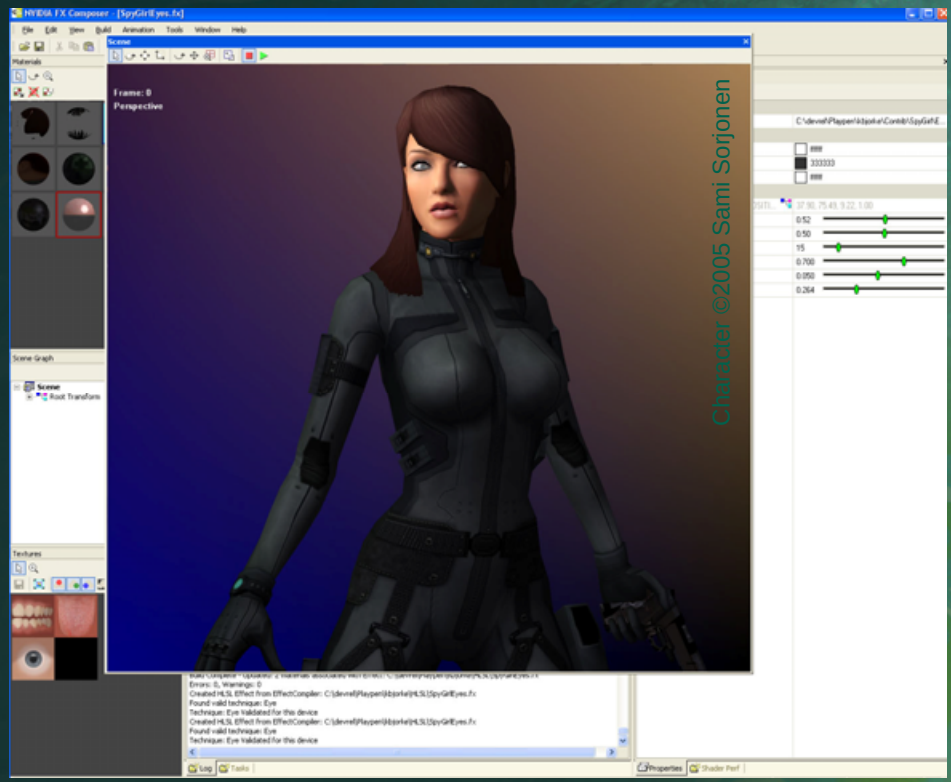
SpyGirl Shaders - Face

- Skin: “subsurface scatter” simulation through diffuse rolloff of (N·L)
  - Similar to Dawn etc
- Eyes: glossy highlight through threshholding
  - Wide spec with a cutoff

Red Subsurface Skin Shading & “Wet” Eye Shaders

SpyGirl Shaders - Suit

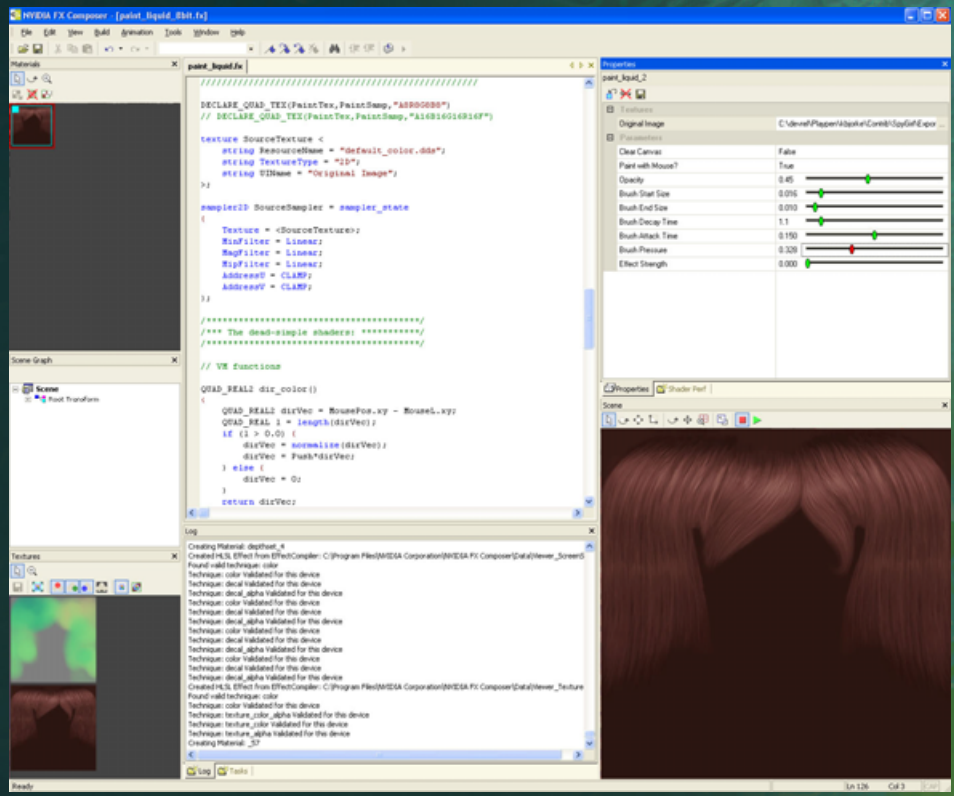
- Adding color changes via “Facing Ratio” gives more solidity to the suit, makes the material seem more complex
- Color-change can also be calculated in the vertex shader, essentially for free



Output from FX Composer

SpyGirl - Hair

- ***Here's Where We Break a Rule***
  - We need an extra asset
- For hair Aniso, we need a direction map
- We can paint one using the "paint_liquid.fx" shader
- FX Composer can save any texture on the fly

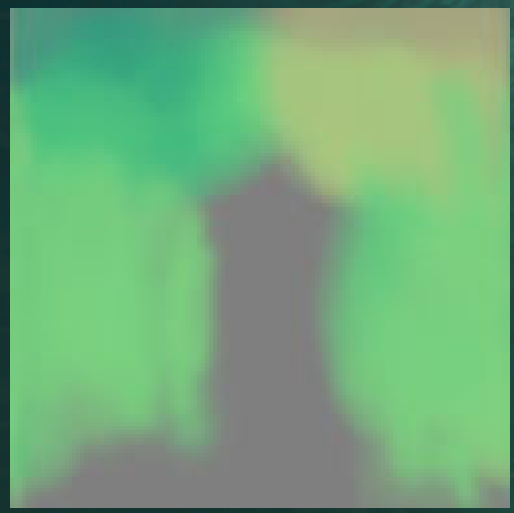


Using "Liquid" to make a hair-direction map

MINI-DEMO

Hair dir map

- Resize it in Photoshop (doesn’t need to be big) and make sure edges are seamless



Direction Map

Color Map

Applying Kajiya-Kay to Hair Surface

- We use tangent/binormal + direction texture to determine the direction of hair threads

Typical Highlight Color

Cyan Highlight for Clarity of Effect

Additional Hair Effects

- We add a soft (N·L) rolloff, much like the skin (but no extra sub-surface color), to emulate some light scattering in the hair
- We add a second rolloff *just for the highlights* for translucent sheen

Integrating Into Production

- Pushing for Consistency Across Environments

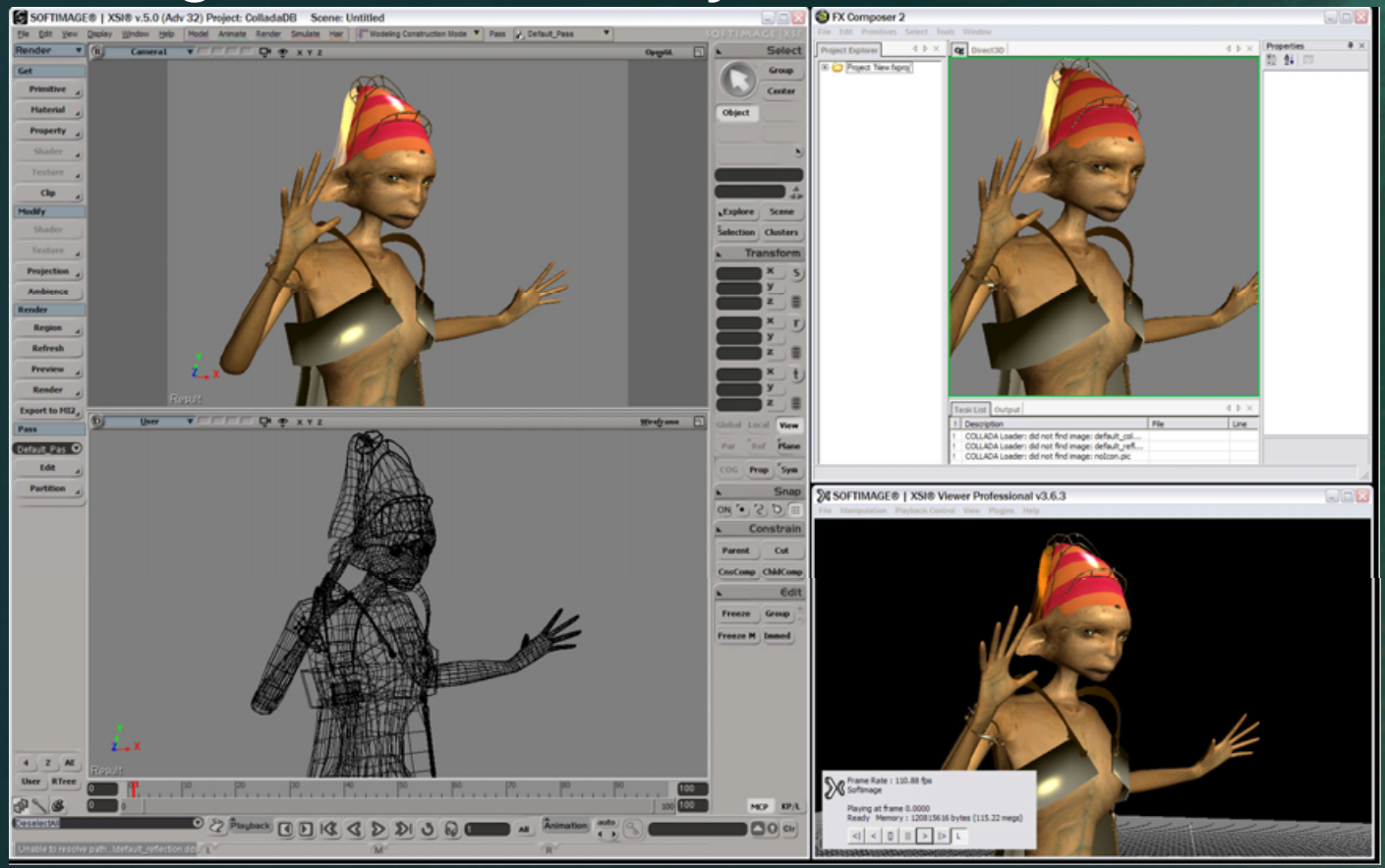
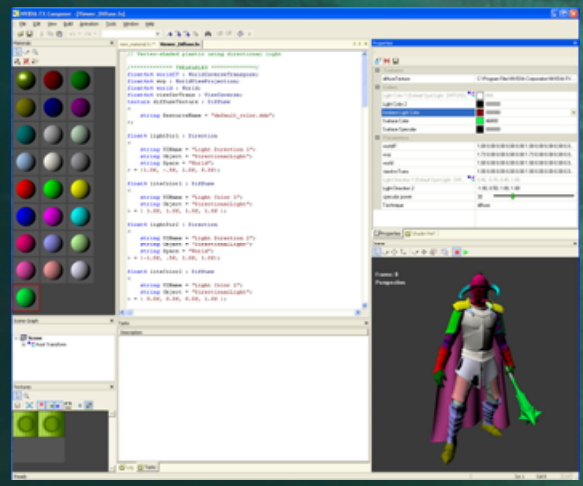


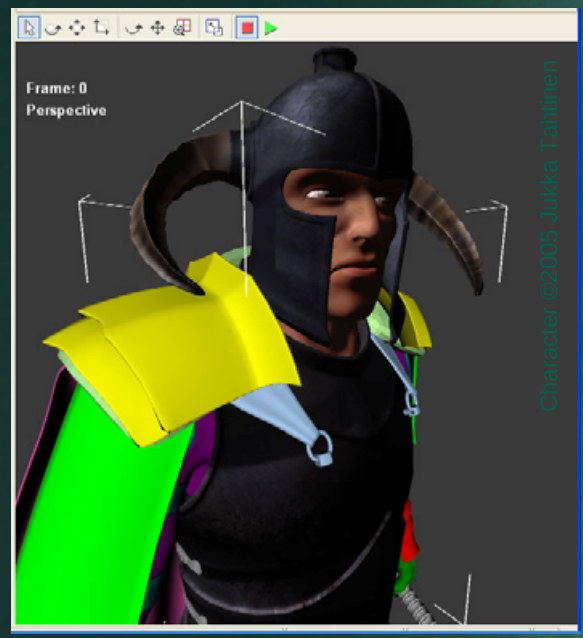
Image Courtesy Jean-Claude Alexandre, SoftImage

XSI 5 and FX Composer 2,0 show simultaneous shader matching via Collada Interface

Wrapup

- Applicable to any model
- Fight CPU-Bound Limitations!





"Warrior" from Jukka Tahtinen, Finland
info@graph3d.com