



*N*VIDIA®

Cg Production Workflow

General Comments

- Use slightly shorter sentences
- Use a color to **highlight** key words/concepts
- Cg language constructs should come **BEFORE** looking at the code samples

Cg User Workflow: An Introduction

- **Making Cg Work For You in Your Applications**
- **Using NVIDIA tools**
- **This Module**
 - **Review of Cg Tools**
 - **Cg with DCC Skills in Mind**
 - **Demonstration**
 - **Exercises**

SW Shading Pipelines

- **The RenderMan Pipeline**

- **Into the RenderMan API:**

- **Dice, Shade, and then Rasterize**

- **In the Shader:**

```
displacement_shader(list_of_light_shaders);  
surface_shader(list_of_light_shaders);  
atmosphere_shader(list_of_light_shaders);
```

- **Other programs (esp. MentalRay) may have even more shader types:**

- **Contour, lens, photon, etc.**

Process Shift: GPU Pipeline

- **The Innovation of the GPU Pipeline**
- **Different Division of Labor for Pipeline**
 - **First:**
 - **Vertex Buffer, Vertex Shader, Rasterize, then Fragment Shader writes to Pixel**
 - **Second:**
 - **Vertex Shaders and Fragment Shaders only**
- **Real-Time Results!**

Why Not RenderMan SL for Real-Time?

- **RenderMan doesn't offer real-time possibilities in the general case.**
- **Hardware-aligned languages mean higher-performance shaders.**
- **Shading-Language differences prove to be key to real-time development.**

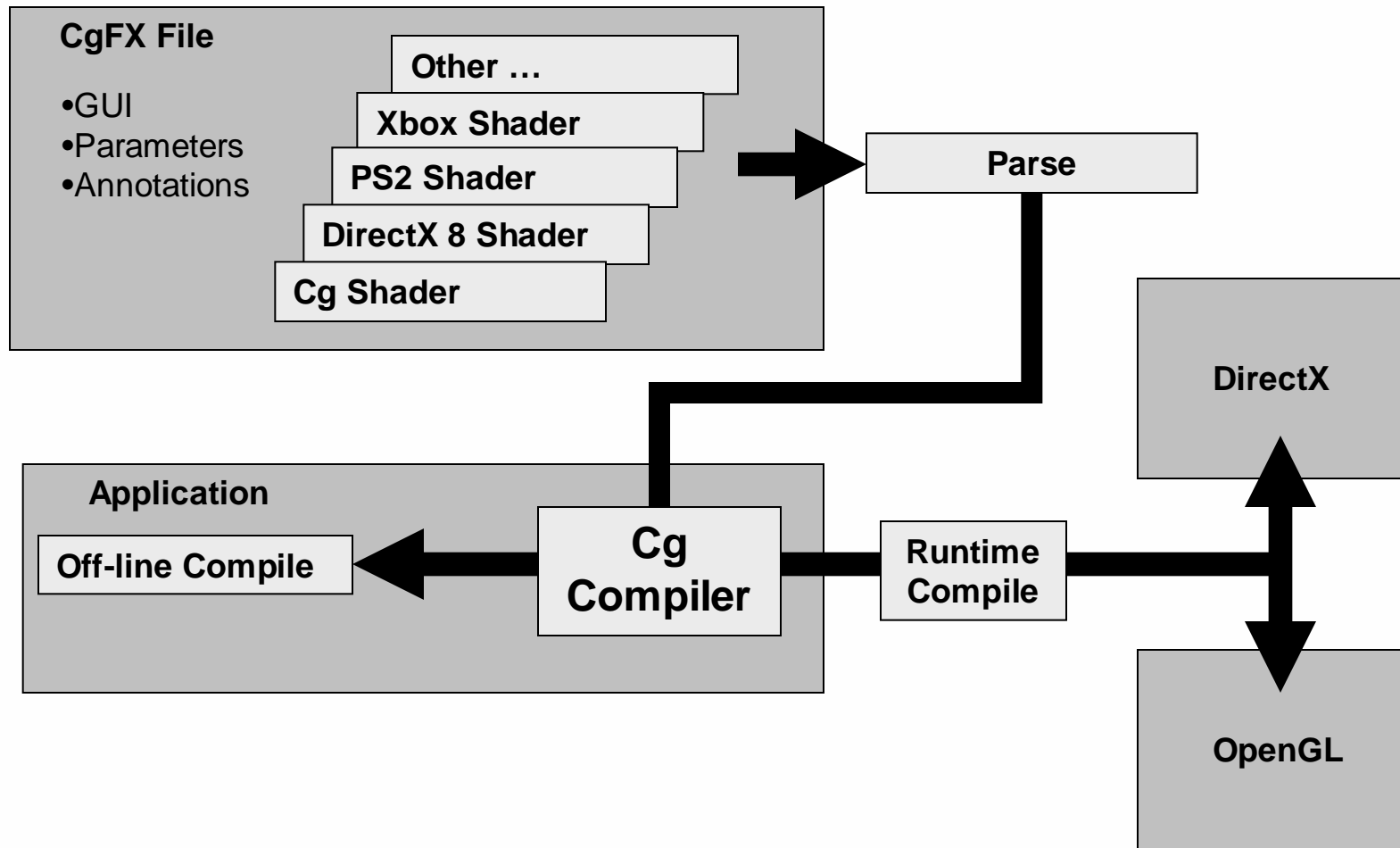
Passes and Combining Shader Types

- **Vertex and Fragment Shaders**
- **RenderMan's algorithm works well with one pass.**
- **More complex graphics require innovation:**
 - **Moving operations to the vertex shader**
 - **Manipulating the overall OpenGL / DirectX graphics state**

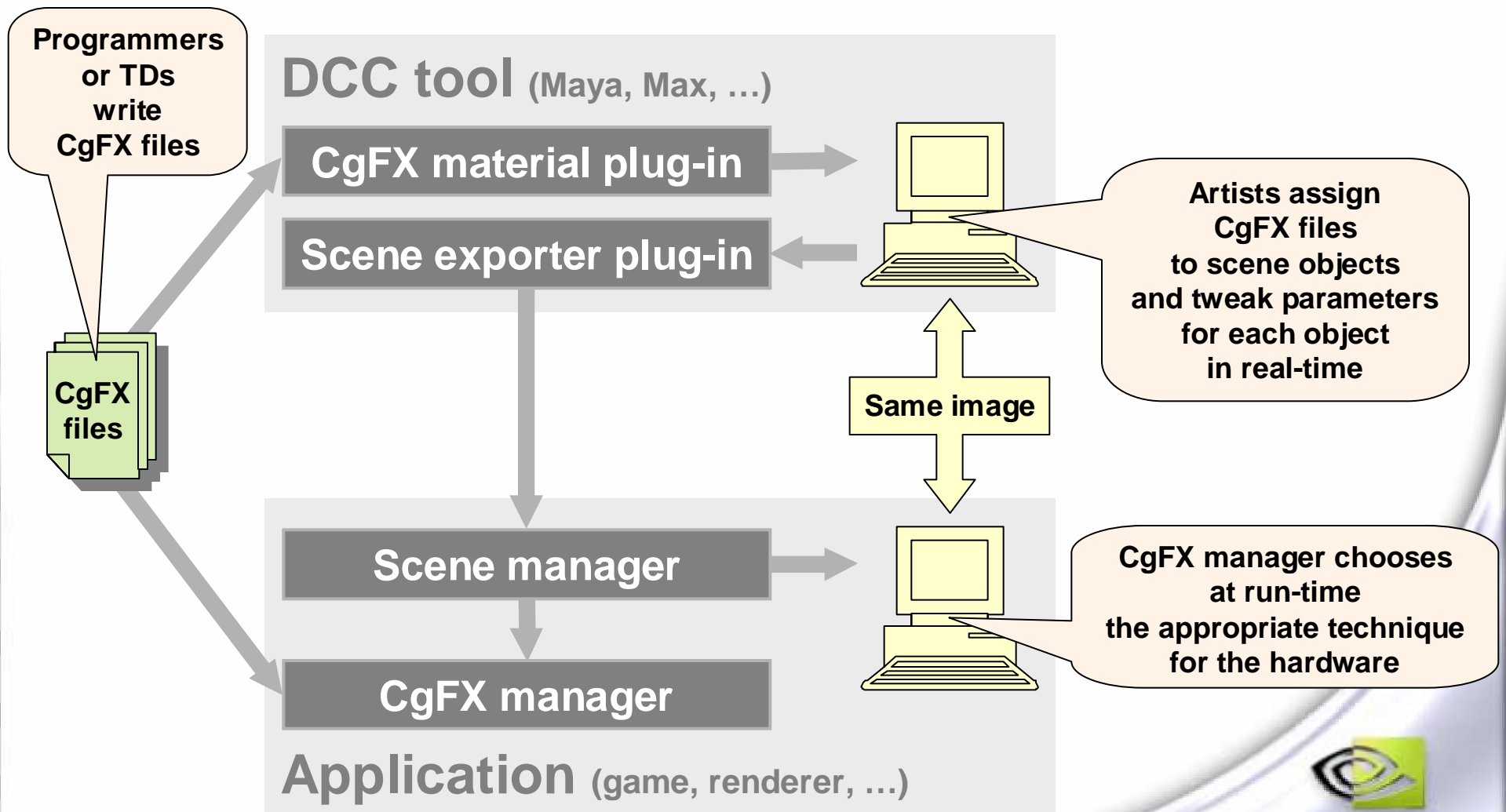
CgFX: A Complete Shading Language

- **A simple way to unify Cg shaders into complete appearances**
- **Can be used at all stages of production**
- **Multiple render passes also supported**
- **Accommodates different hardware devices with distinct rendering capabilities**

CgFX in a Standard Application



CgFX Production Pipeline



CgFX Example File - Structure

- 1. Global declarations**
- 2. Tweakables/Tracking Declarations**
- 3. Vertex Shaders**
- 4. Pixel Shaders**
- 5. Techniques to encapsulate Shaders, Tweakables, Render Passes & Graphics-State Settings.**

CgFX Example - Globals

```
/******NVMH3*****/
Comments:
    Cg Class Template Shader
    Bumpy, fresnel-shiny, dielectric, textured, one quadratic point source
    + ambient.

Mini-Style Guide:
    Shader parameter names start with Caps.
    Connector member names start with Caps.
    FX tweakable names start with lower case letters.
    Local shader variable names generally start with lower case letters.
    L, P, N, V are typical Light vector, Point, Normal, View.
    Vectors that are normalized have names that end in "n" (except "Nb").
    When possible use float3 calculations to free-up w terms for compiler.
    Float4 tweakables (such as light color) may be float3 in the shader. Take care.
    Always provide a DX8 fallback technique, even if it doesn't look
        quite right -- it'll be available as a fast path or for users
        with lesser computers.
*****/

string description = "Siggraph 2003 Cg Class Shader Template";
string Category = "Template";
string keywords = "bumpmap,texture,glossmap,fresnel,quadratic,pointlight";
```

CgFX Example - UnTweakables

```
/****** "UN-TWEAKABLES," TRACKED BY CPU APPLICATION *****/  
  
float4x4 worldIT : WorldIT;  
float4x4 wvp : WorldViewProjection;  
float4x4 world : World;  
float4x4 viewIT : ViewIT;
```

CgFX Example – Data Structures

```
/* data from application vertex buffer */
struct appdata {
    float3 Position : POSITION;
    float4 UV       : TEXCOORD0;
    float4 Normal   : NORMAL;
    float4 Tangent  : TEXCOORD1;
    float4 Binormal : TEXCOORD2;
};

/* vertex shader output */
struct vertexOutput {
    float4 HPosition      : POSITION;
    float2 TexCoord       : TEXCOORD0;
    float3 PtLightVec     : TEXCOORD1;
    float3 WorldNormal    : TEXCOORD2;
    float3 WorldView      : TEXCOORD3;
};
```

CgFX Example – Vertex Shader

```
vertexOutput basicVS(appdata IN,  
    uniform float4x4 WorldViewProj,  
    uniform float4x4 WorldIT,  
    uniform float4x4 World,  
    uniform float4x4 ViewIT,  
    uniform float3 LightPos    // in world coordinates  
) {  
    vertexOutput OUT;  
    OUT.WorldNormal = mul(WorldIT, IN.Normal).xyz;  
    float4 Po = float4(IN.Position.xyz,1.0);    // object coordinates  
    float3 Pw = mul(World, Po).xyz;            // world coordinates  
    OUT.PtLightVec = LightPos - Pw;           // world coordinates  
    OUT.TexCoord = IN.UV.xy;  
    OUT.WorldView = normalize(ViewIT[3].xyz - Pw);    // obj coords  
    OUT.HPosition = mul(WorldViewProj, Po);    // screen clipspace coords  
    return OUT;  
}
```

CgFX Example – Pixel Shader

```
float4 basicPS(vertexOutput IN,
    uniform sampler2D ColorMap,
    uniform samplerCUBE EnvMap,
    uniform float Kd,
    uniform float Ks,
    uniform float SpecExpon,
    uniform float Metalness,
    uniform float Kr,
    uniform float KrMin,
    uniform float FresExp,
    uniform float3 AmbiColor,
    uniform float3 SurfColor,
    uniform float3 PtLightColor,
    uniform float PtIntensity
) : COLOR {
    float3 colorTex = SurfColor * tex2D(ColorMap,IN.TexCoord).xyz;
    float3 Nn = normalize(IN.WorldNormal);
    float3 Vn = normalize(IN.WorldView);
    float ld = 1.0 / length(IN.PtLightVec);
    float3 Ln = ld * IN.PtLightVec;          // normalizes
    float3 Hn = normalize(Vn + Ln);
    float hdn = dot(Hn,Nn);
    float ldn = dot(Ln,Nn);
    float4 litVec = lit(ldn,hdn,SpecExpon);
    ldn = (ld*litVec.y) * PtIntensity;      // multiply by (1/dist) twice for quadratic falloff
    float3 diffContrib = ld*(ldn * PtLightColor);
    float3 specContrib = ld*((ldn * litVec.z * Ks) * PtLightColor);
    float3 reflVect = reflect(Vn,Nn);
    float vdn = dot(Vn,Nn);
    float fres = KrMin + (Kr-KrMin) * pow((1.0-abs(vdn)),FresExp);
    float3 reflColor = lerp(fres,Kr,Metalness) * texCUBE(EnvMap,reflVect).xyz;
    float3 result = (colorTex*(Kd*diffContrib+AmbiColor)) +
        lerp(((1.0).xxx),colorTex,Metalness)*(specContrib + reflColor);
    return float4(result.xyz,1.0);
}
```

CgFX Example – DirectX8 Fallback

```
struct vertexOutput8 {
    float4 HPosition          : POSITION;
    float2 TexCoord : TEXCOORD0;
    float4 DiffColor         : COLOR0;
    float4 SpecColor         : COLOR1;
};

vertexOutput8 fallbackVS(appdata IN,
    uniform float4x4 WorldViewProj,
    uniform float4x4 WorldIT,
    uniform float4x4 World,
    uniform float4x4 ViewIT,
    uniform float3 AmbiColor,
    uniform float3 SurfColor,
    uniform float Kd,
    uniform float Ks,
    uniform float SpecExpon,
    uniform float3 LightPos,          // in world coordinates
    uniform float3 PtLightColor,
    uniform float PtIntensity
) {
    vertexOutput8 OUT;
    float3 Nn = normalize(mul(WorldIT, IN.Normal).xyz);
    float4 Po = float4(IN.Position.xyz,1.0);          // object coordinates
    float3 Pw = mul(World, Po).xyz;                  // world coordinates
    float3 Vn = normalize(ViewIT[3].xyz - Pw);        // obj coords
    float3 L = LightPos - Pw;
    float Ld = 1.0/dot(L,L);
    float3 Ln = normalize(L);
    float3 Hn = normalize(Vn + Ln);
    float hdn = dot(Hn,Nn);
    float ldn = dot(Ln,Nn);
    float4 litVec = lit(ldn,hdn,SpecExpon);
    ldn = (Ld*litVec.y) * PtIntensity;
    float3 diffContrib = AmbiColor + (ldn * PtLightColor);
    float3 specContrib = (ldn * litVec.z) * PtLightColor;
    OUT.DiffColor = float4(diffContrib.xyz,1.0);
    OUT.SpecColor = float4(specContrib.xyz,1.0);
    OUT.TexCoord = IN.UV.xy;
    OUT.HPosition = mul(WorldViewProj, Po);          // screen clipspace coords
    return OUT;
}

float4 fallbackPS(vertexOutput8 IN, uniform sampler2D ColorMap) : COLOR
{
    float3 colorTex = IN.DiffColor.xyz * tex2D(ColorMap,IN.TexCoord).xyz;
    float3 result = colorTex + IN.SpecColor.xyz;
    return float4(result.xyz,1.0);
}
```

CgFX Example – Light Tweakables

```
float4 lightPos : Position
<
    string Object = "PointLight";
    string Space = "World";
> = {100.0f, 100.0f, 100.0f, 1.0f};

float4 lightColor : Specular <
    string Desc = "Light Color";
> = {1.0f, 1.0f, 1.0f, 1.0f};

float lightIntensity
<
    string gui = "slider";
    float uimin = 1.0;
    float uimax = 10000.0;
    float uistep = 1.0;
    string Desc = "lamp power";
    float min = 0.0;
    float max = 10000.0;
> = 10.0;

//////////////////////////////////////// ambient light

float4 ambiLightColor : Ambient
<
    string Desc = "Ambient Light Color";
> = {0.07f, 0.07f, 0.07f, 1.0f};
```

CgFX Example – Surface Tweakables

```
float4 surfColor : Diffuse
<
    string Desc = "Surface Color";
> = {1.0f, 1.0f, 1.0f, 1.0f};

float diffStrength
<
    string gui = "slider";
    float uimin = 0.0;
    float uimax = 1.5;
    float uistep = 0.01;
    string Desc = "Kd";
    float min = 0.0;
    float max = 1.5;
> = 1.0;
```

CgFX Example – Texture Tweakables

```
texture colorTexture : DiffuseMap
<
    string File = "default_color.dds";
    string TextureType = "2D";
>;

sampler2D colorSampler = sampler_state
{
    Texture = <colorTexture>;
    MinFilter = Linear;
    MagFilter = Linear;
    MipFilter = Linear;
};
```

CgFX Example – CUBE Texture

```
texture envTexture : EnvMap
<
  string File = "default_reflection.dds";
  string TextureType = "Cube";
>;

samplerCUBE envSampler = sampler_state
{
  Texture = <envTexture>;
  MinFilter = Linear;
  MagFilter = Linear;
  MipFilter = Linear;
  AddressU = clamp;
  AddressV = clamp;
  AddressW = clamp;
};
```

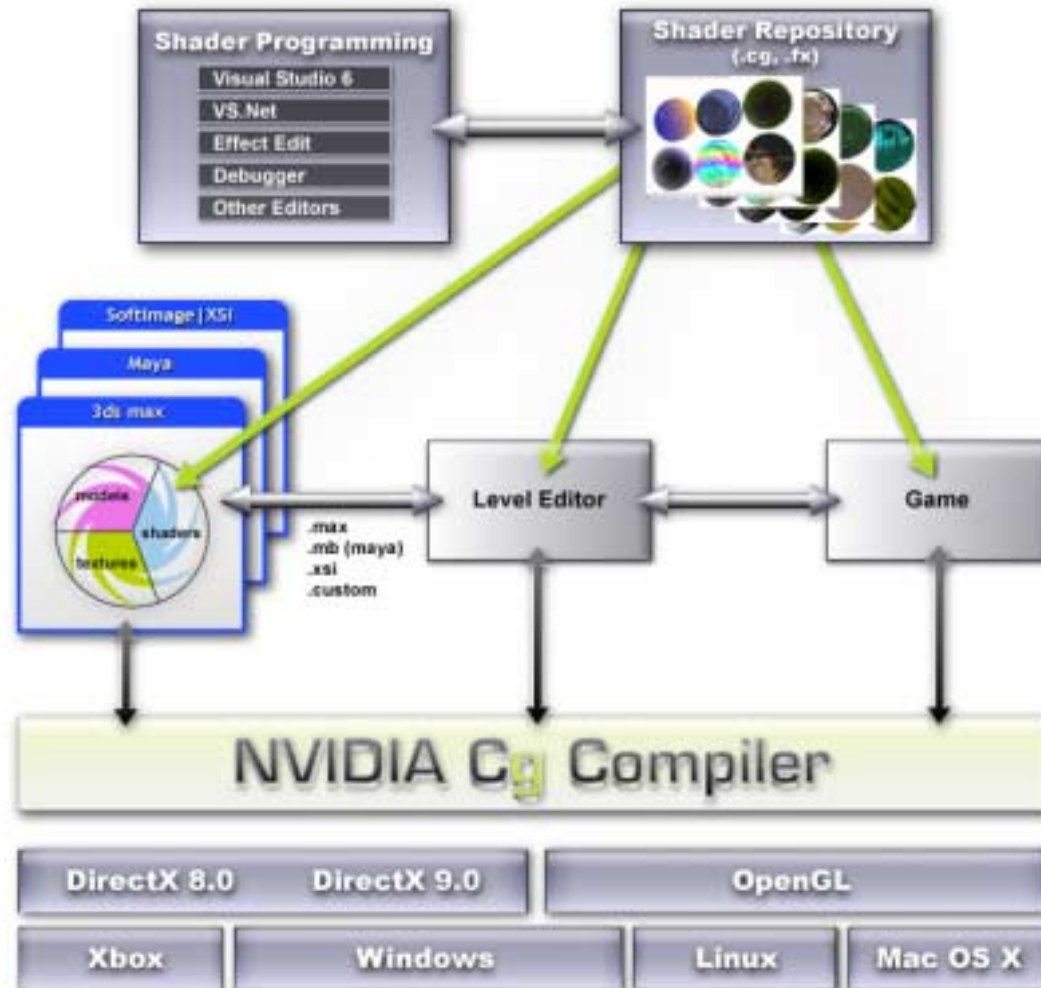
CgFX Example – Technique

```
technique Main
{
    pass p0
    {
        VertexShader = compile vs_2_0 basicVS(wvp,
            worldIT,world,viewIT,lightPos);
        ZEnable = true;
        ZWriteEnable = true;
        CullMode = None;
        PixelShader = compile ps_2_x basicPS(colorSampler,
            envSampler,diffStrength,
            specStrength,specExpon,metalness,
            reflStrength,reflMin,fresExp,
            ambiLightColor,surfColor,
            lightColor,lightIntensity);
    }
}
```

CgFX Example – Alternate Technique

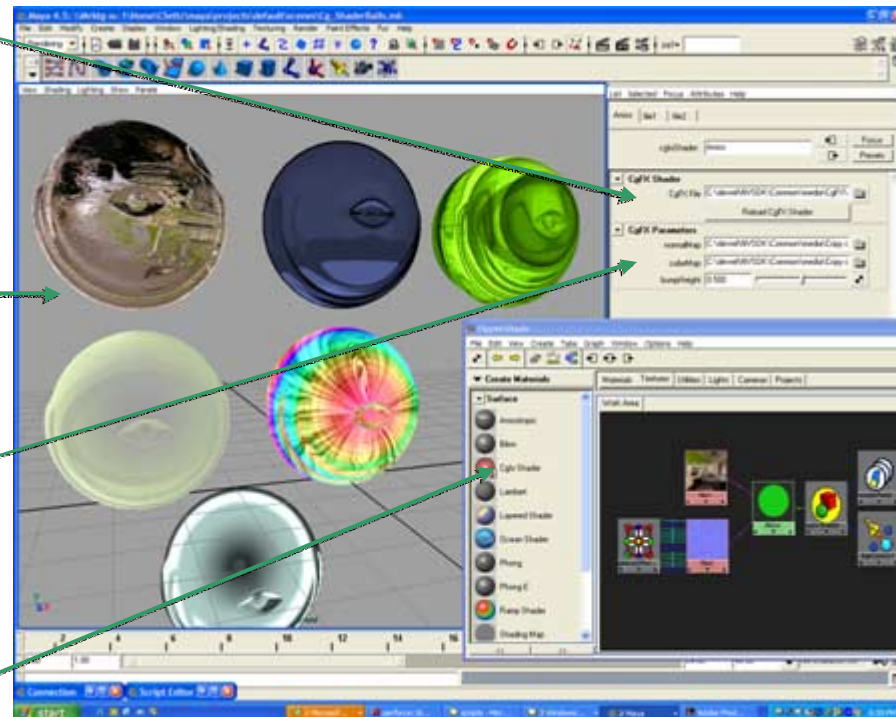
```
technique DirectX8
{
    pass p0
    {
        VertexShader = compile vs_1_1 fallbackVS(wvp,
            worldIT,world,viewIT,
            ambiLightColor,surfColor,
            diffStrength,specStrength,specExpon,
            lightPos,lightColor,lightIntensity);
        ZEnable = true;
        ZWriteEnable = true;
        CullMode = None;
        PixelShader = compile ps_1_1 fallbackPS(colorSampler);
    }
}
```

Creating Frictionless Workflow



Maya Cg Plug-in

- **Dynamic Shader specific GUI**
 - Supports standard GPU image formats, e.g., .dds
 - Supports .fx shader format
- **Samples include:**
 - Bumpy Shiny, Toon, Anisotropic Metal, Ghostly, Refraction Dispersion, Rainbow
- **Integrated with Maya's lights**
- **Intuitive artist controls**
 - Slider control over key real-time parameters (e.g., bump depth)
- **CgFX integrated with Maya's hypershade – node based shader editor**



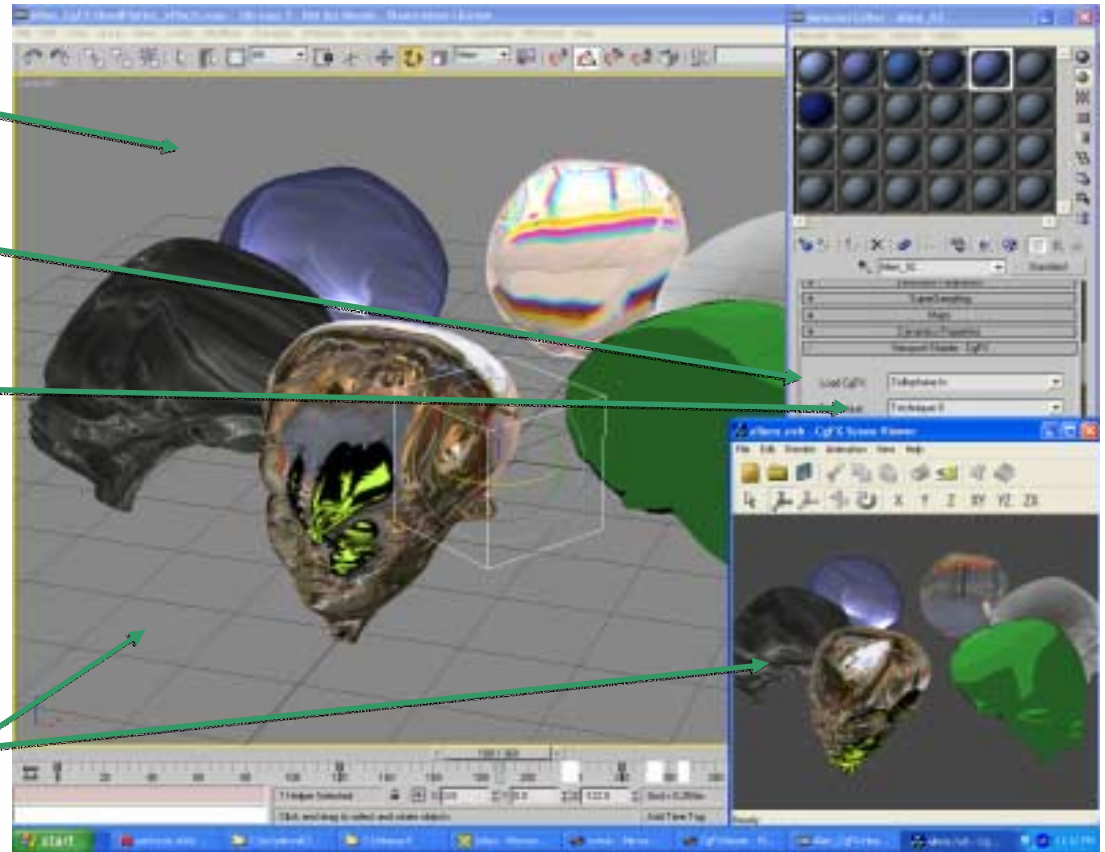
Maya Cg Tools to Accelerate Workflow

- **Written in Mel for Easy Customization**
- **Dedicated “CgFX” Tool Shelf**
- **Web-based Help throughout**
- **User-Centered Functions Include:**
 - Select Shaders based not just on cgfxShader nodes, but by any object or poly face selection
 - Edit .fx shaders
 - Build Instant Test Scenes
 - Assign Keyable Attributes
 - Auto-Convert Maya shaders to CgFX
 - Auto-Connect Lighting Attributes
 - Create Shared clickable control objects for fast UI
 - Reload Shared Shaders *en masse*
 - Add shaders to existing Maya surfaces
 - Dedicated Shader-Editing Windows
 - Toggle Wireframe rendering per-object



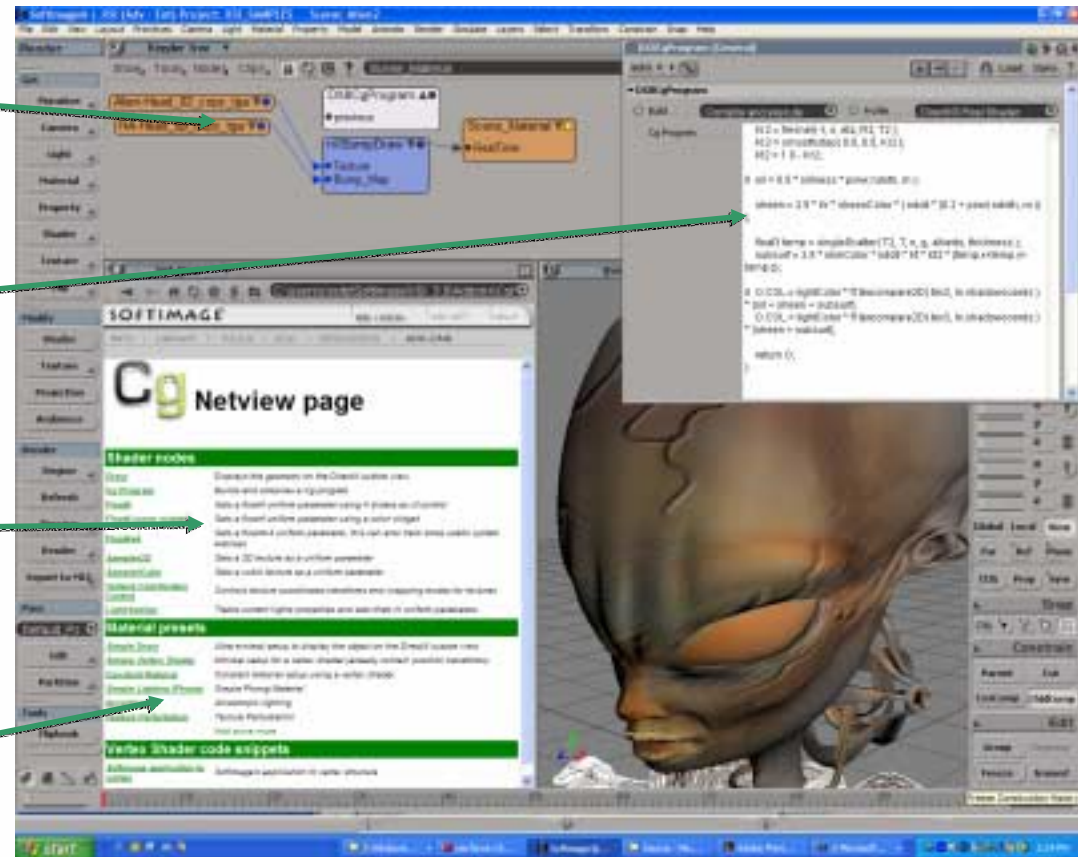
3ds max Cg Plugin

- CgFX Viewport manager enables I/O of real-time shaders
- Supports .fx shader format
- Select multiple techniques for shader fallbacks
- Intuitive artist controls (Sliders, Color Choosers...)
- Dynamic Shader specific GUI
- DirectX or OpenGL display



Softimage|XSI

- Cg integration in XSI's Render Tree
- Direct Cg code editing and compilation
- Net View for help, samples & documentation
- Interactive shader builder
- Shipping with XSI 3.0 and later



Cross-platform NVSDK: Linux and Mac

- Linux & Macintosh updates through CVS

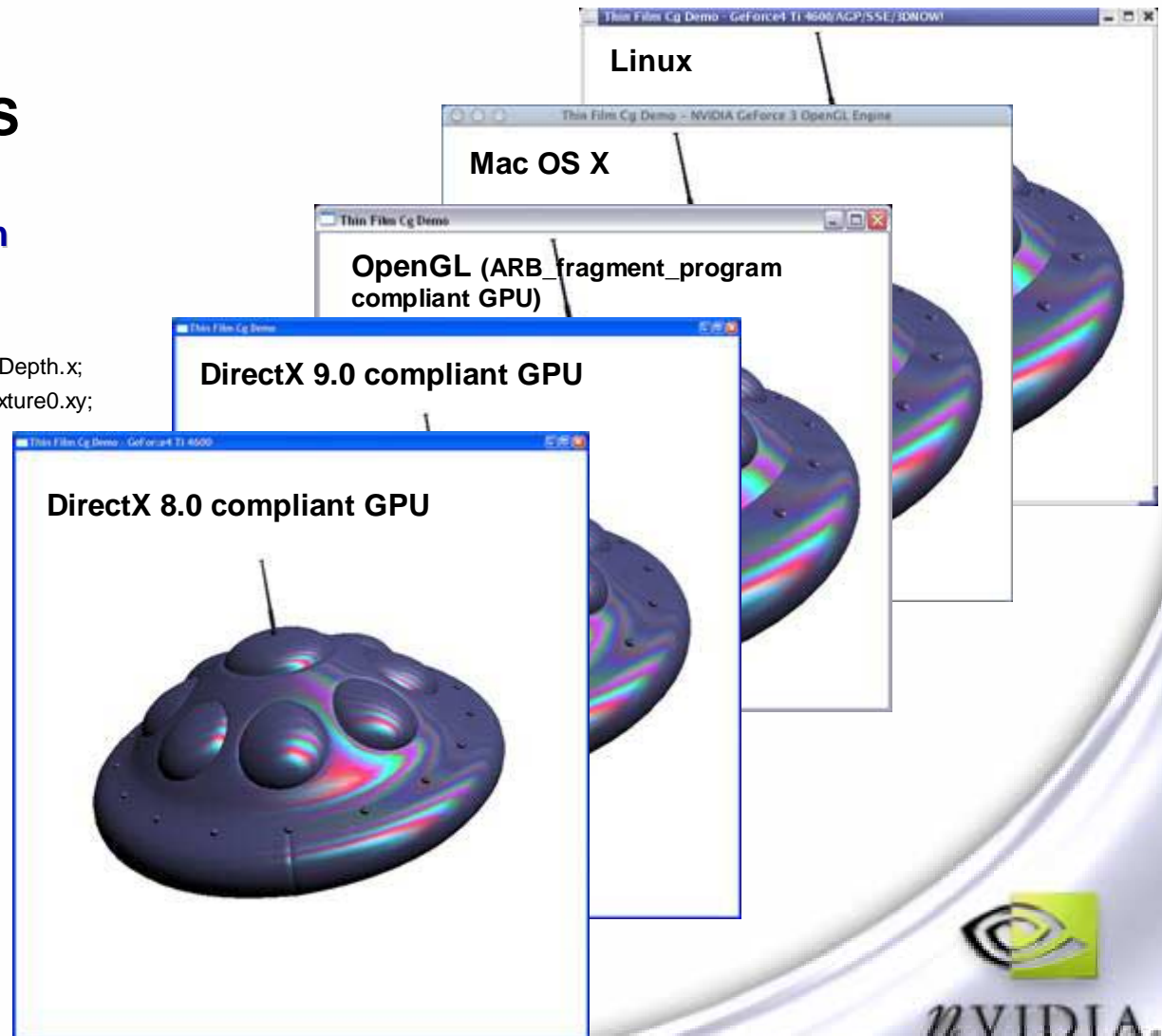
Thin Film Dispersion Cg Program

Vertex Shader

```
float viewdepth = (1.0 / dot(normalVec, eyeVec)) * FilmDepth.x;  
OUT.filmDepth = viewdepth.xx; OUT.diffCoord = IN.Texture0.xy;  
float3 lightVec = normalize((float3)LightVector);  
float3 halfAngleVec = normalize(lightVec + eyeVec);  
float diffuse = dot(normalVec, lightVec);  
float specular = dot(normalVec, halfAngleVec);  
float4 lighting = lit(diffuse, specular, 32);  
OUT.diffCol = (float4)lighting.y;  
OUT.specCol = (float4)lighting.z; ...
```

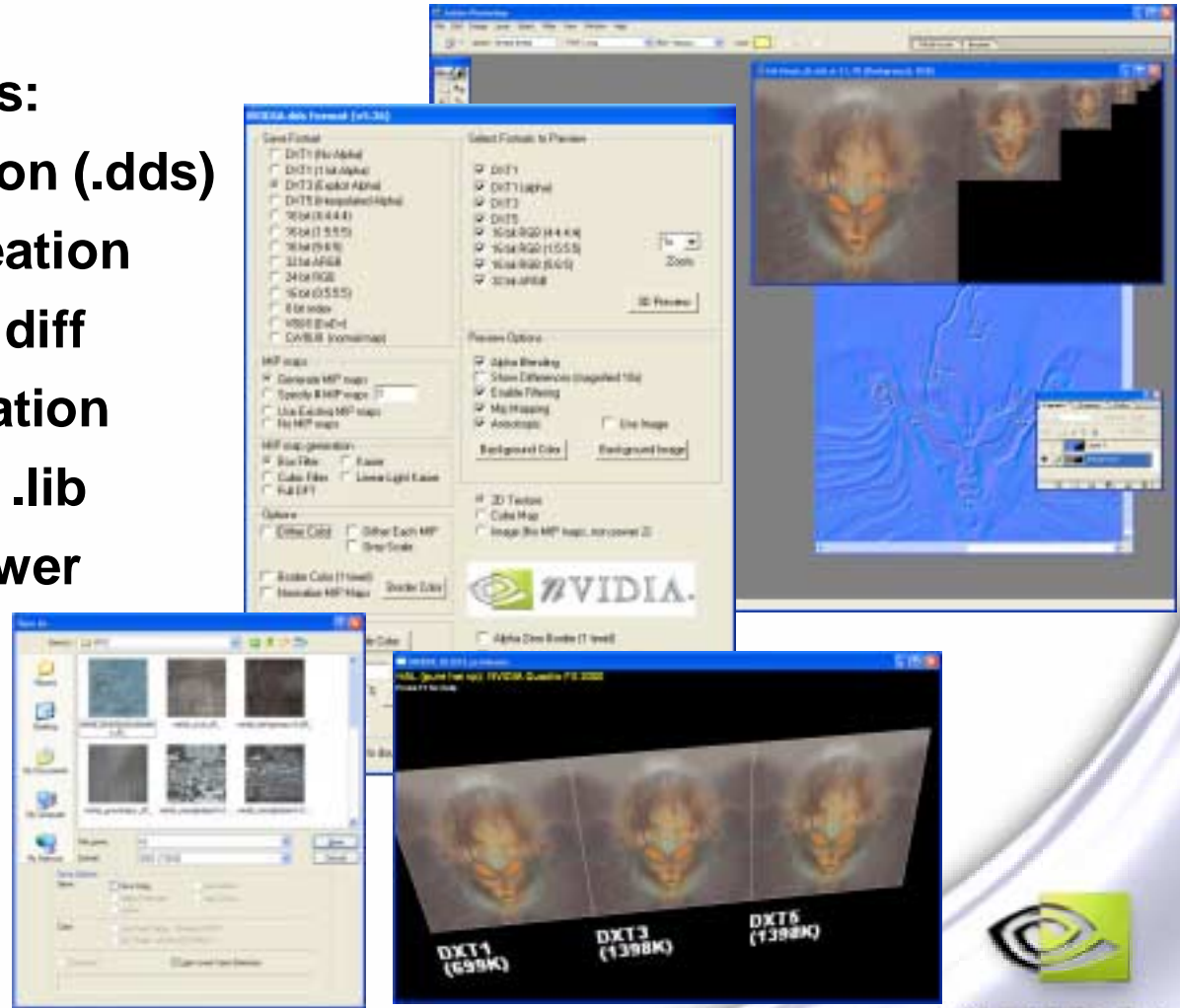
Pixel Shader

```
float4 diffCol = float4(0.1, 0.1, 0.1, 0.1);  
float4 fringeCol = tex2D(fringeMap, IN.filmDepth);  
color = fringeCol*IN.specCol +  
IN.diffCol*diffCol; color.a = 1.0;
```



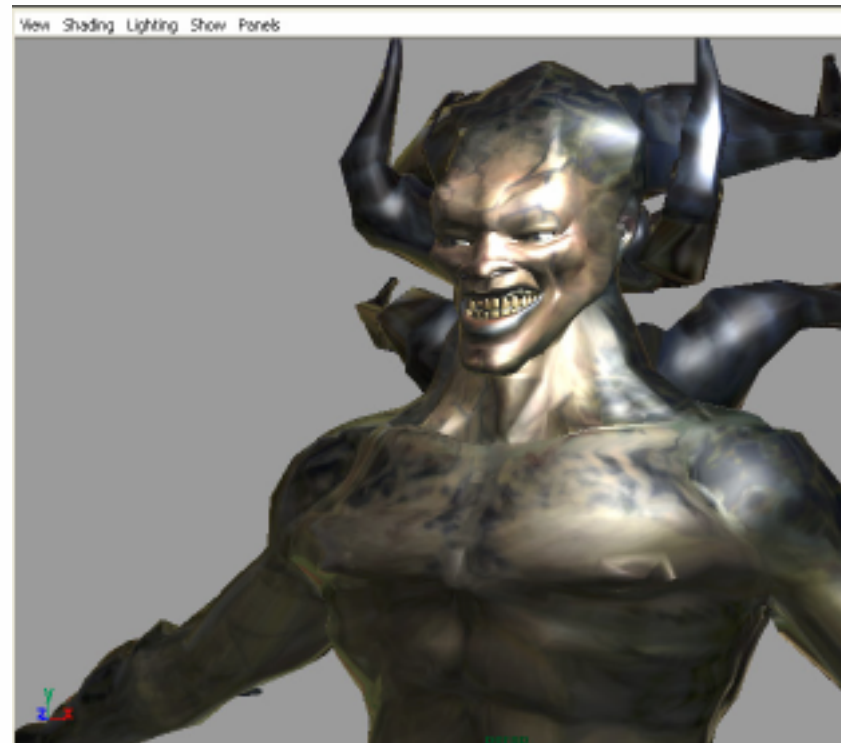
Texture Tools & Plugins

- Photoshop Plug-ins:
 - DXT compression (.dds)
 - Normal Map creation
 - 3d preview and diff
 - MIP map generation
- Command line and .lib
- DDS thumbnail viewer



Exercise 1: Rapid Prototyping

- Use the CgFX Plugin to see what CgFX is doing.
- Key:
 - If you know RenderMan, you will understand CgFX
 - Downloadable browser that you can use to edit an existing CgFX file
- Take a few minutes to gain hands-on experience with CgFX and rapid prototyping



Hands-On Agenda

- **Convert a bumpmap to a hardware-friendly format via PhotoShop, for use in realtime shading**
- **Apply a shader in 3D Studio Max 5.1**
- **See the result in Max – with our bumpmap**
- **Edit the Shader Code from within Max**
- **See the new result in Max**
- **Look at fancier example in Max**
- **Switch to Maya**
- **See the our shader from Max – in Maya!**
- **Complex Maya examples**

The Shader We'll Use: Car/Truck Paint

- Uses BRDF, Normal, and Environment Maps
- BRDF map (provided) to emulate color-shift in paint color
 - Mixed with “traditional” diffuse/specular shading calculations
- Normal map (we'll create one) for bumps
- Env map (provided) for enameled sheen



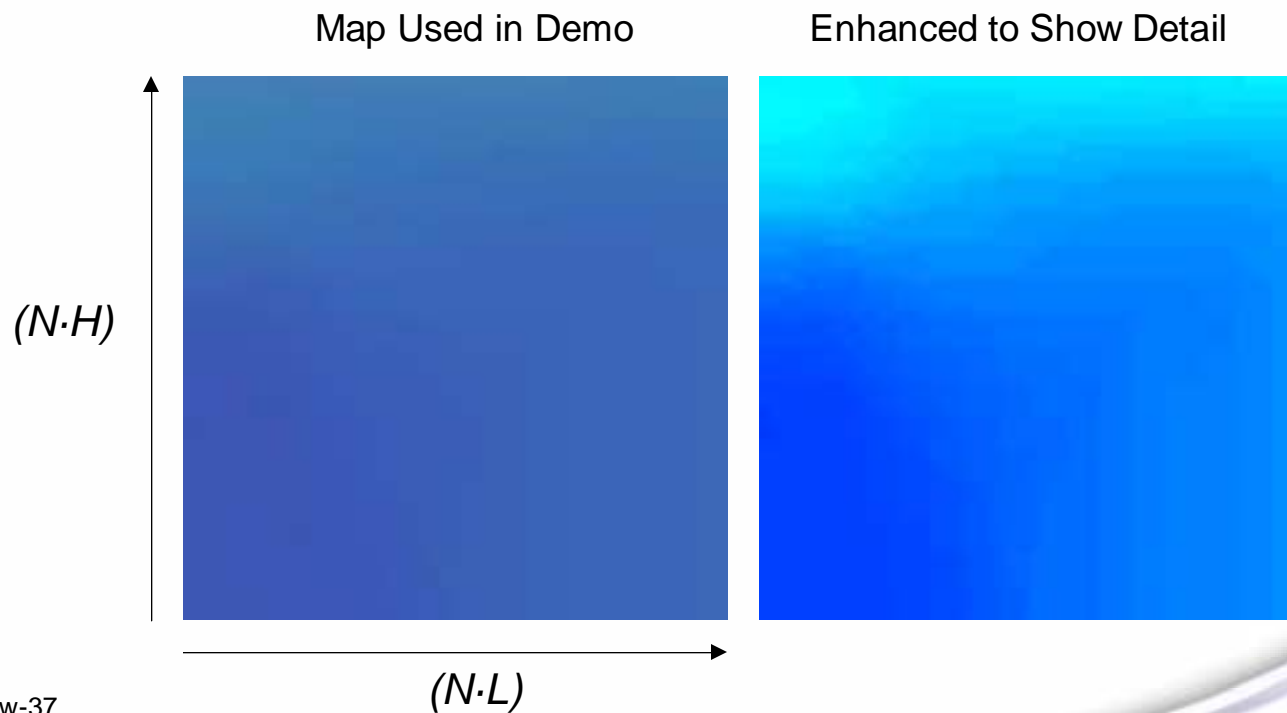
A Texture in Place of Complex Math

- **Texture Maps commonly used instead of math functions – they're often faster and artist-friendly:**
 - **Noise**
 - **BRDFs**
 - **Splines and Gradients (as shown at right)**
 - **Light Falloff**
 - **Normalization CUBE maps**
 - **Dependant Textures**



Our BRDF Texture: Paint Color

- Texture started as a “physically based” BRDF, based on Ford “Cayman” paint
- Then tweaked by an artist in Photoshop to get what he *really* wanted!



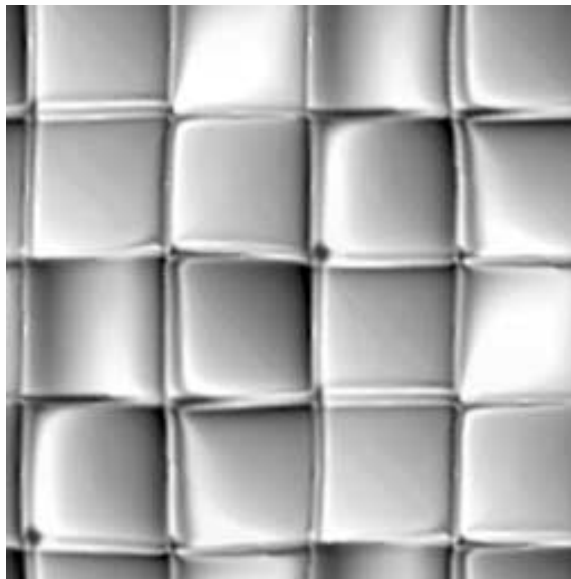
Tangent-Space Normal Maps

- What is “Tangent Space?”
- We all live in the Earth’s Tangent Space (“up” is relative)
- In a tangent-space map, red is “right,” green is “forward,” and blue (if used) is “up”
- When using normal maps instead of bump maps, be aware that you’ve filtered-down some detail to generate normals



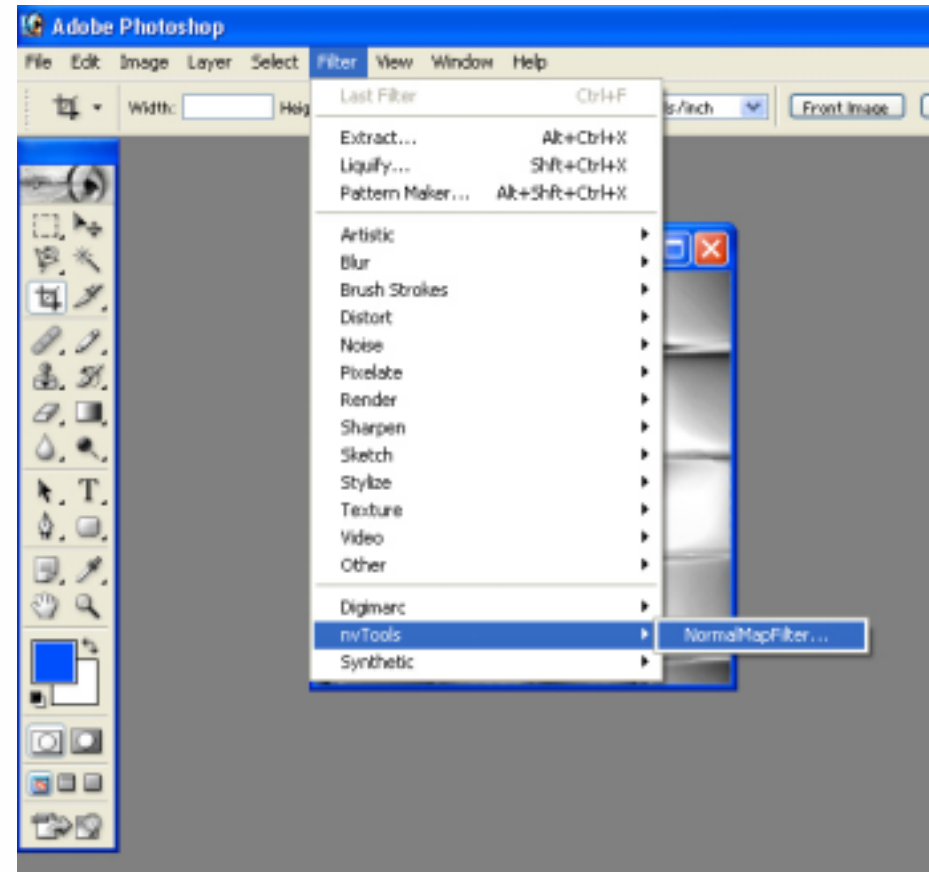
Bumpmaps in Photoshop

- We want a color normal map, not a grayscale bumpmap
- Start PhotoShop
- Open “Buckles.tga”



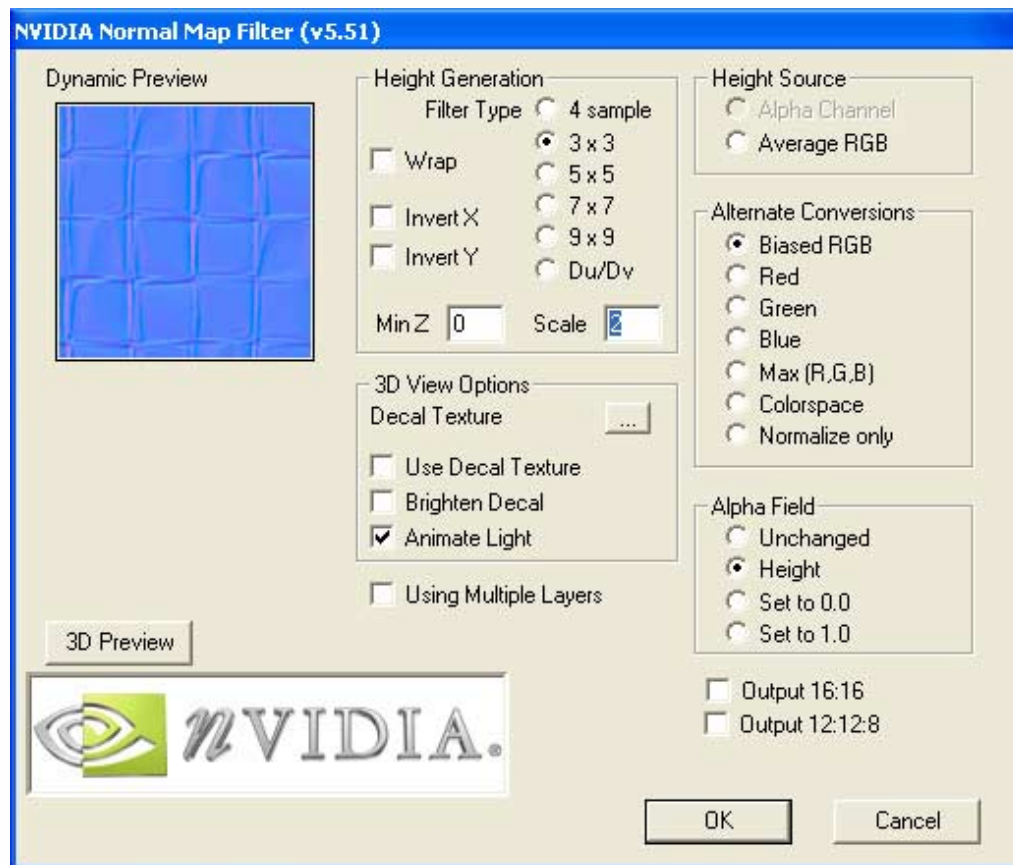
Convert Greyscale to A Normal Map

- **Menu Filter->nvTools->NormalMapFilter...**
- **Ignore “info” popup box – this is typical for RGB-only TGA files**



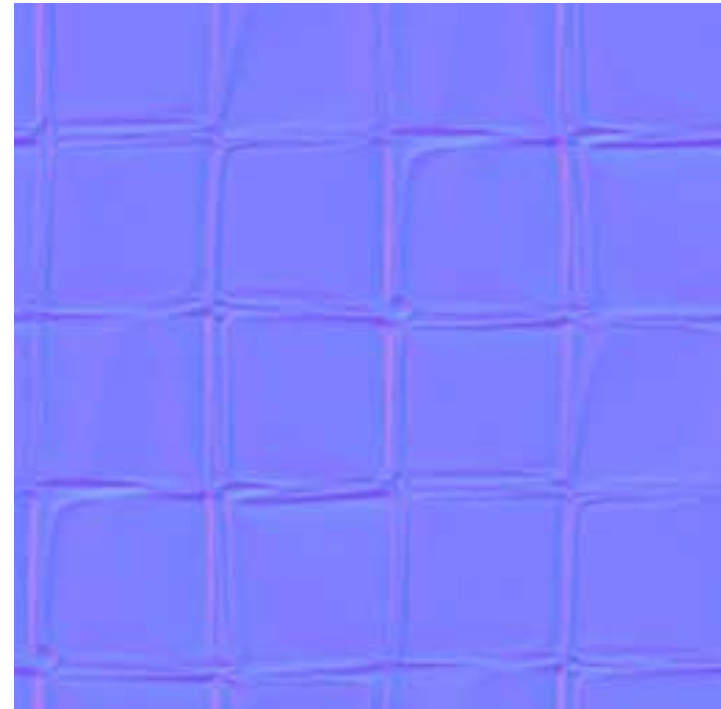
Select Normal-Mapping Options

Visible in Preview Window

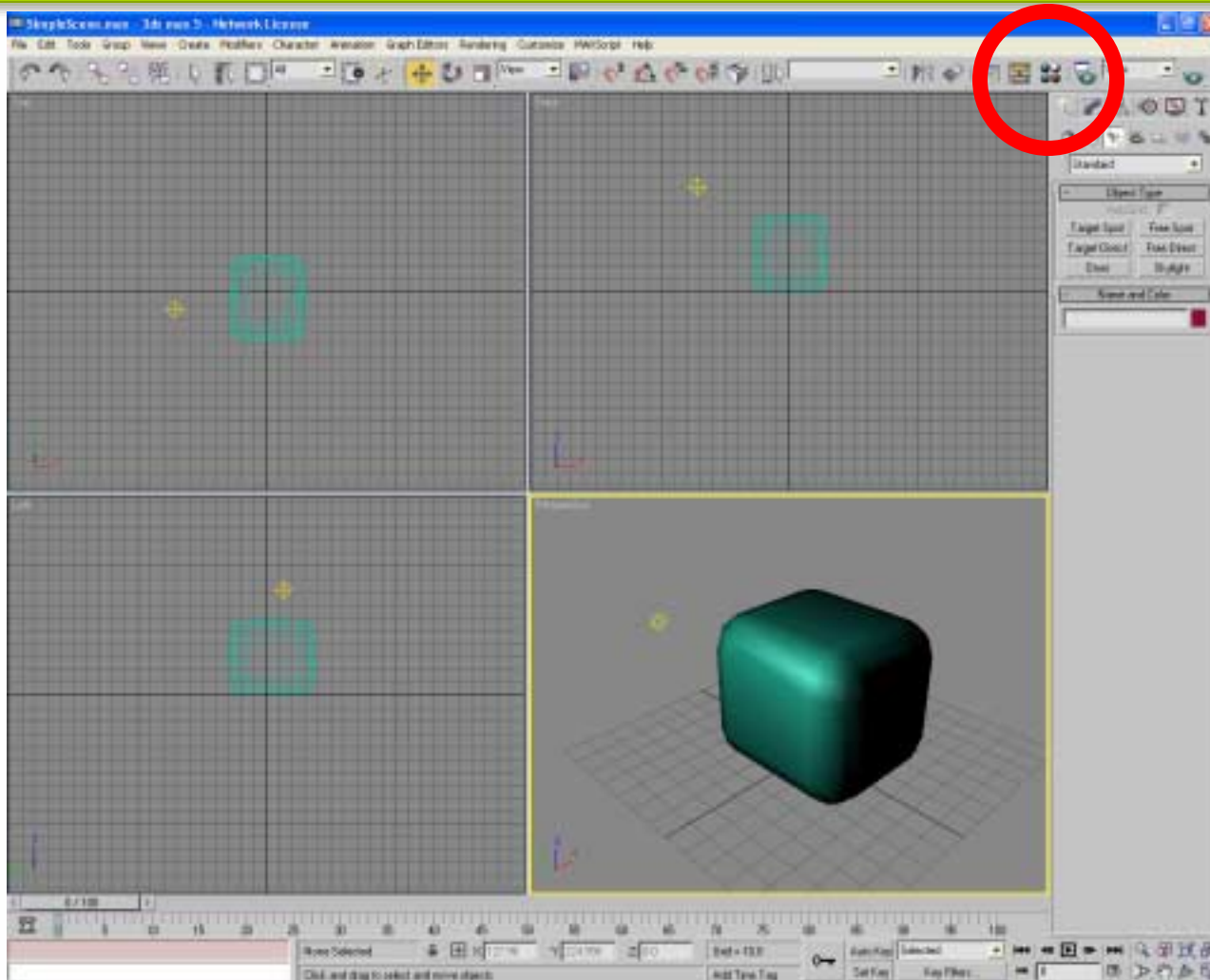


Save to DDS Format

- **File->Save As...**
- **Format: DDS**
- **Accept Defaults:**
 - **DXT1 RGB**
 - **Generate MIP maps**
 - **SAVE**
- **All done, so close PhotoShop**
- **Next Stop: 3ds max 5**

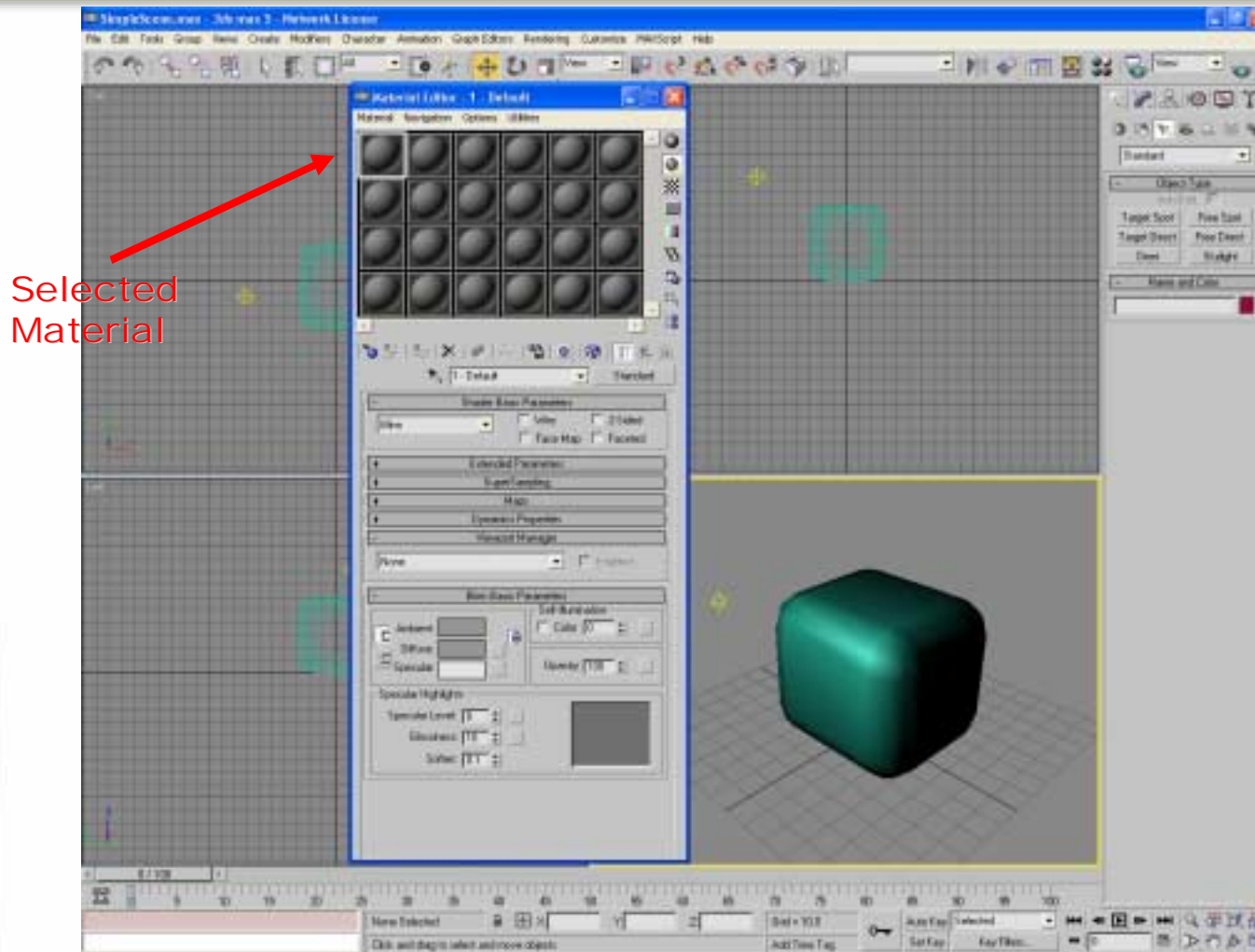


Load “SimpleScene” in Max 5.1

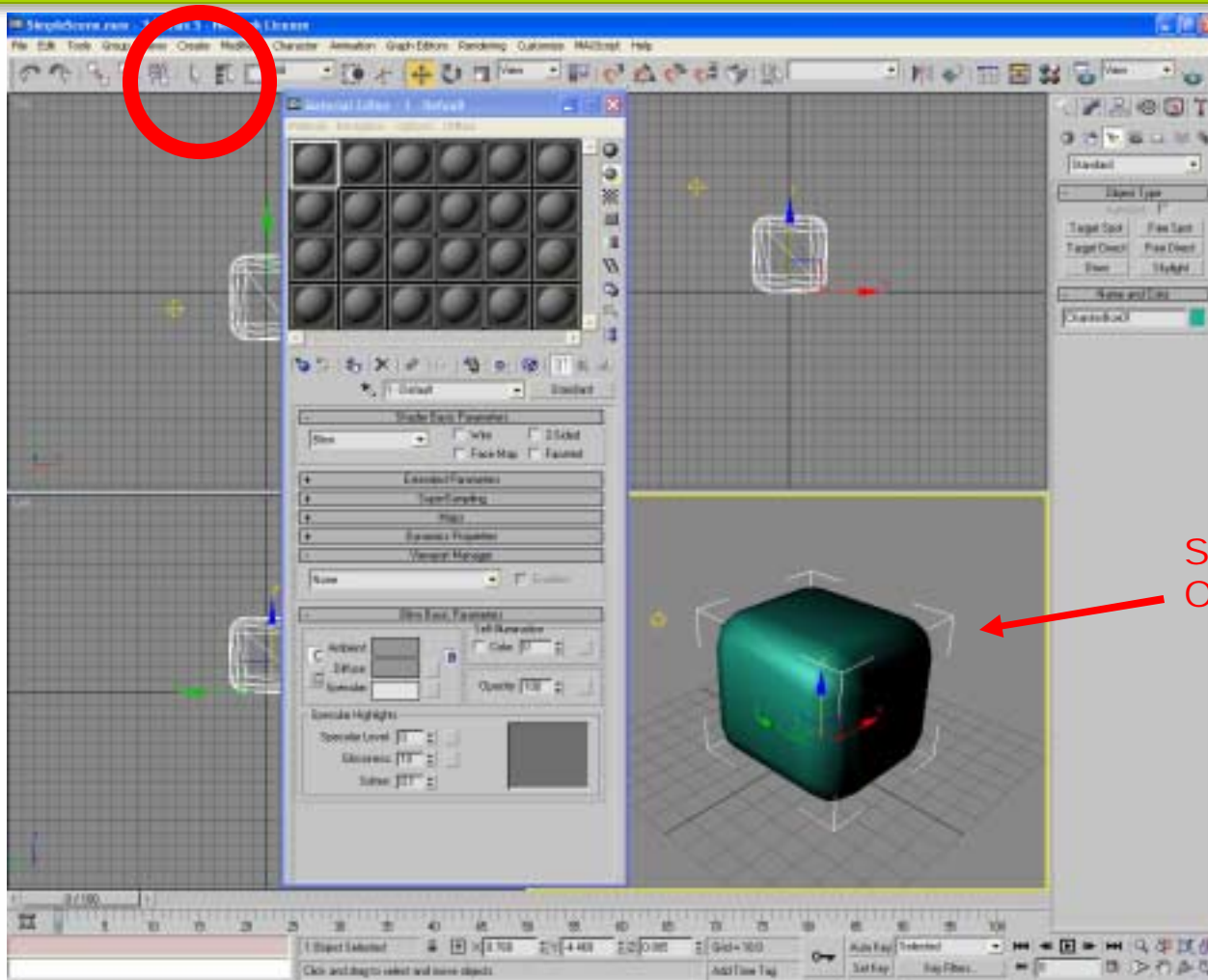


- Simple “chamfer box” and one-light scene
- Press billiard balls to open Materials Editor

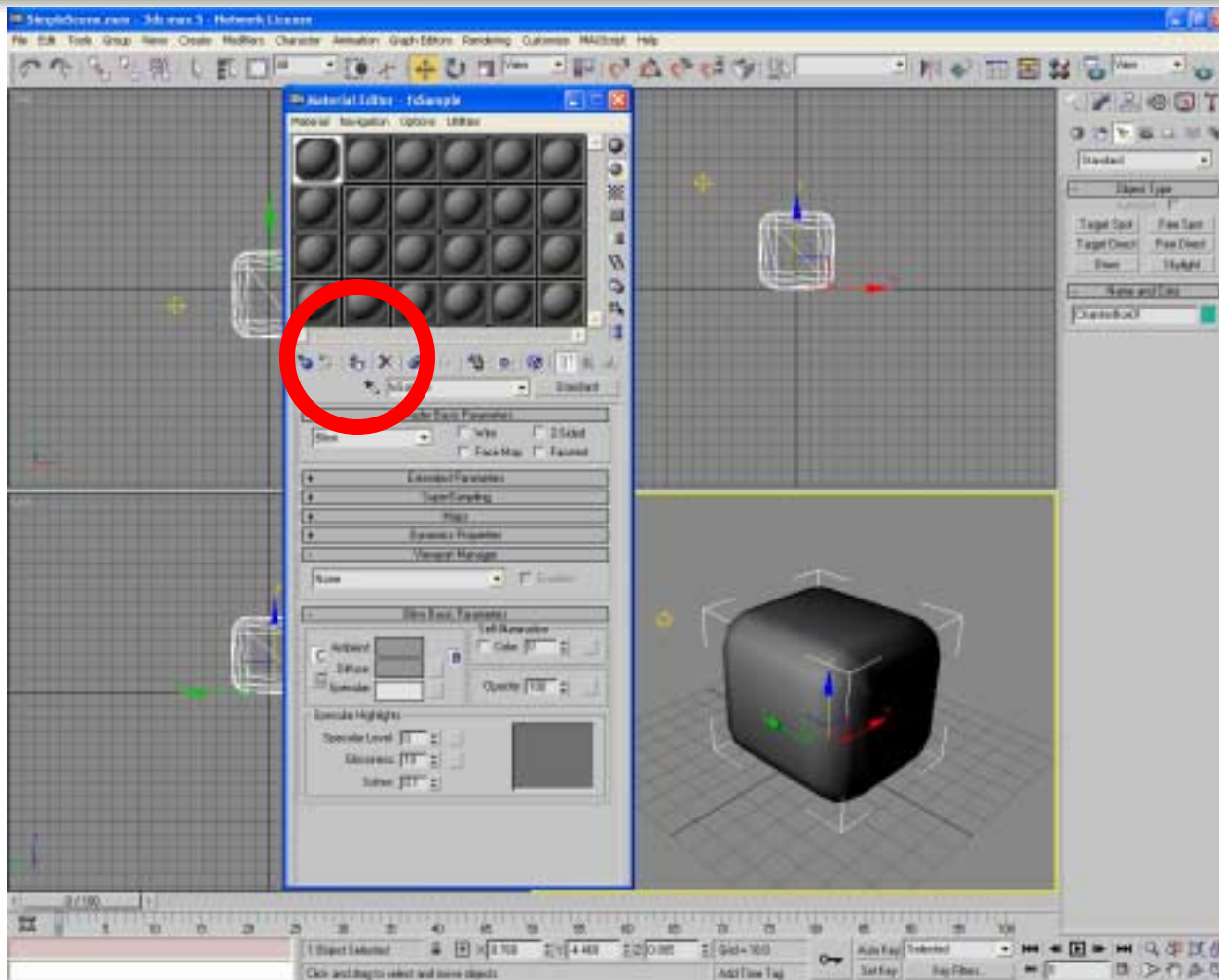
Display Material Editor



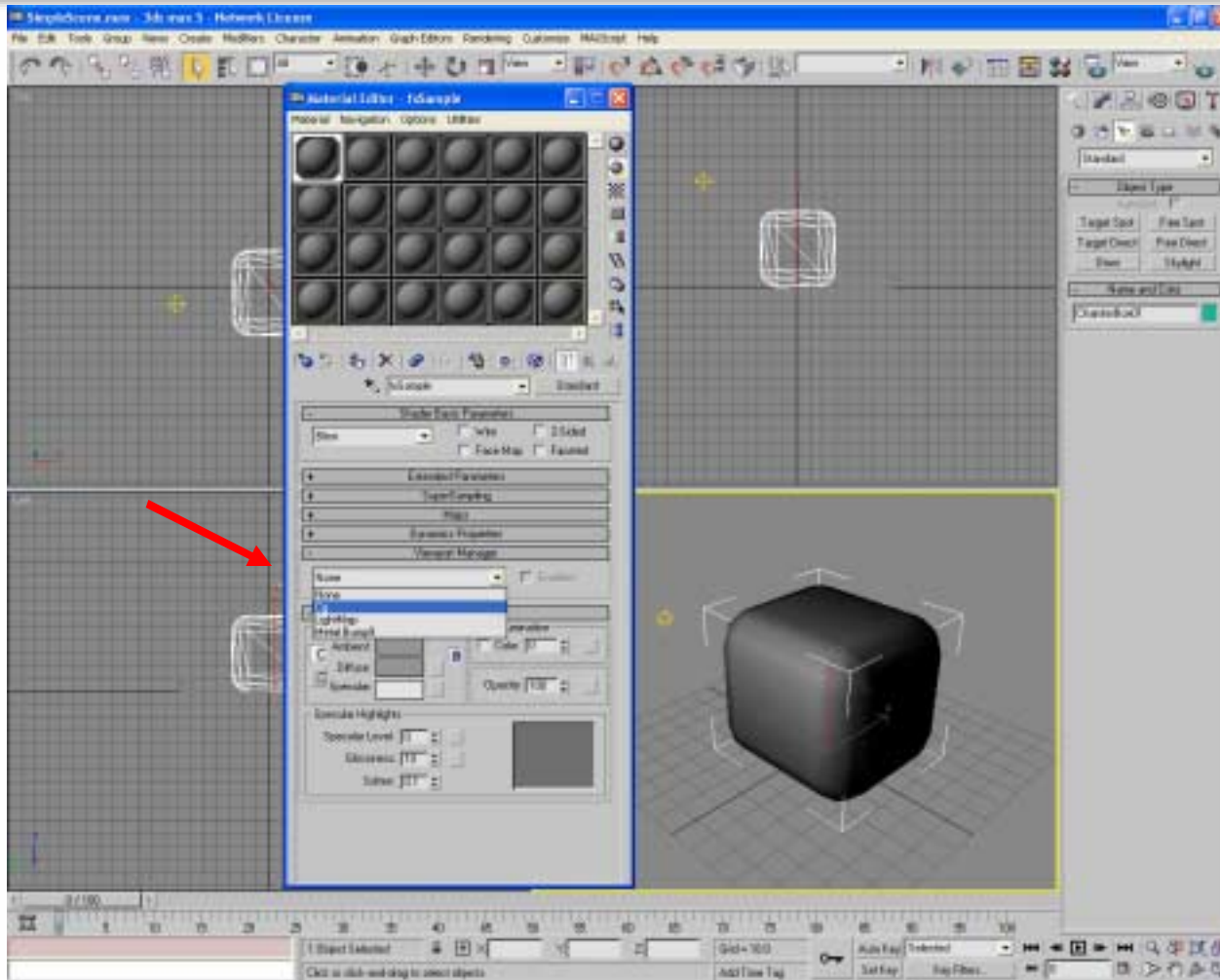
Select Object



Assign Surface To Selected Object

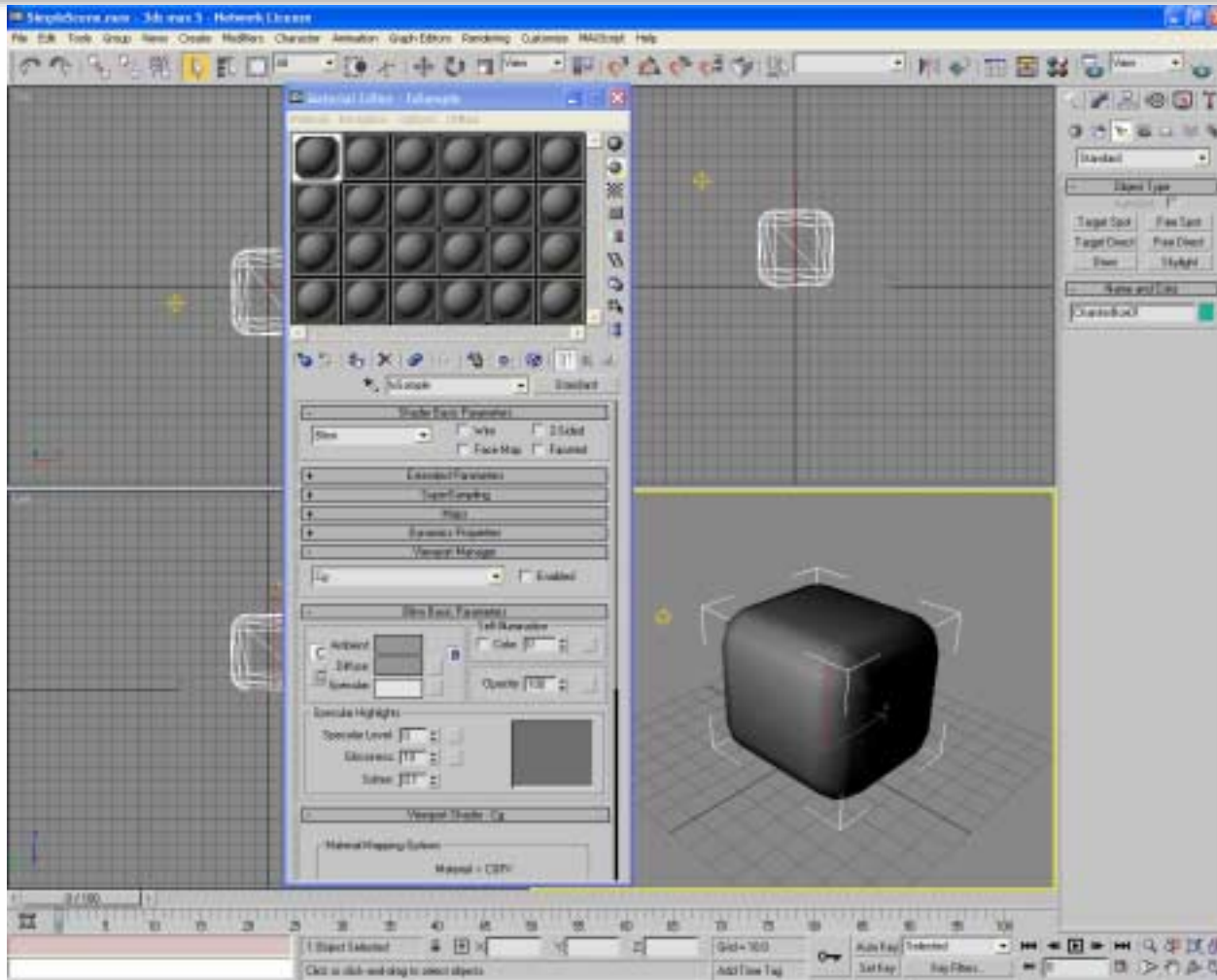


Switch Viewport Manager for This Material



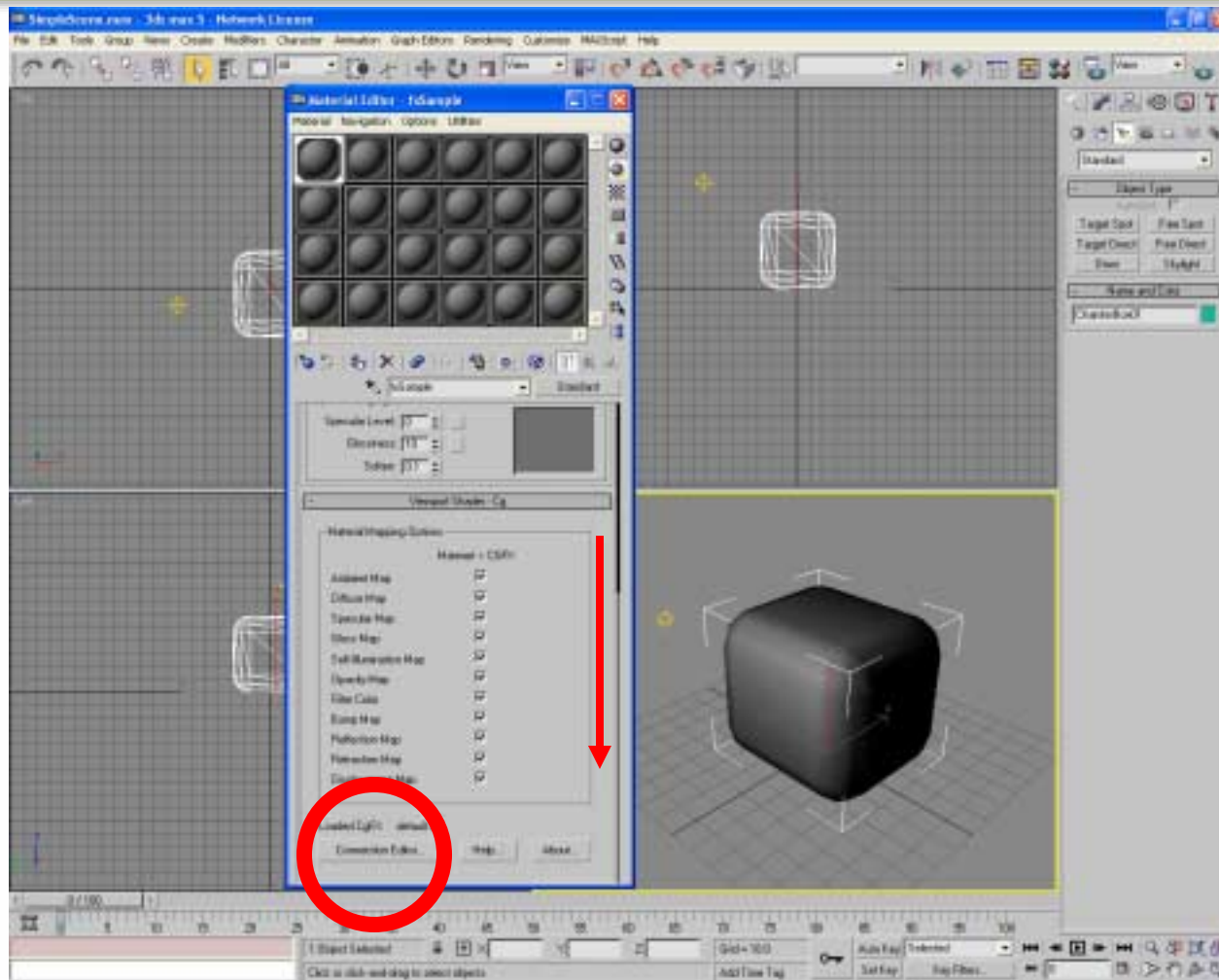
- Sets the manager, but doesn't inherently activate it

Cg Selected – But No Visible Change?



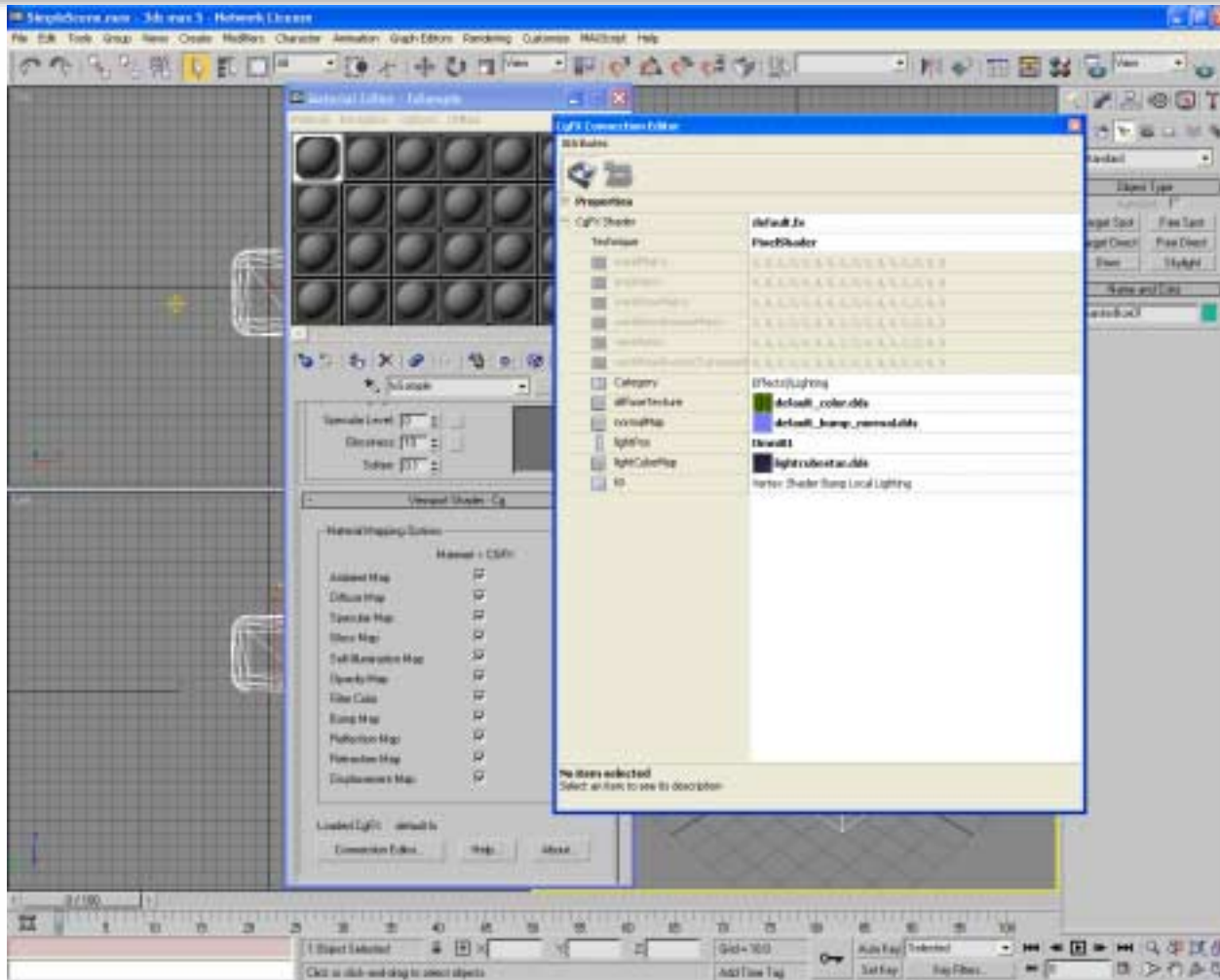
- We'll enable the shader in a minute...

Scroll Down in Material Editor



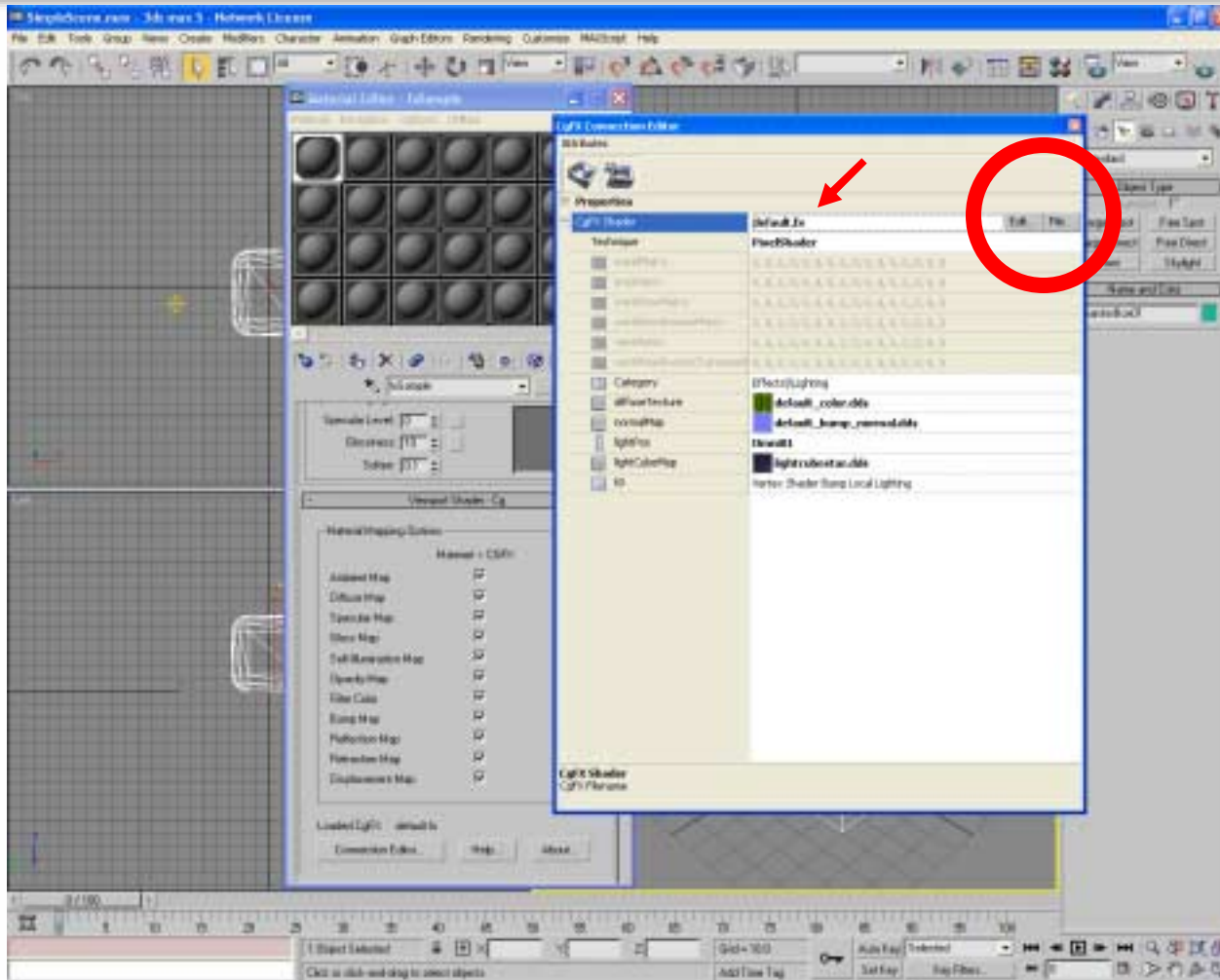
- Look for buttons at the bottom of the window

Click “Connection Editor”



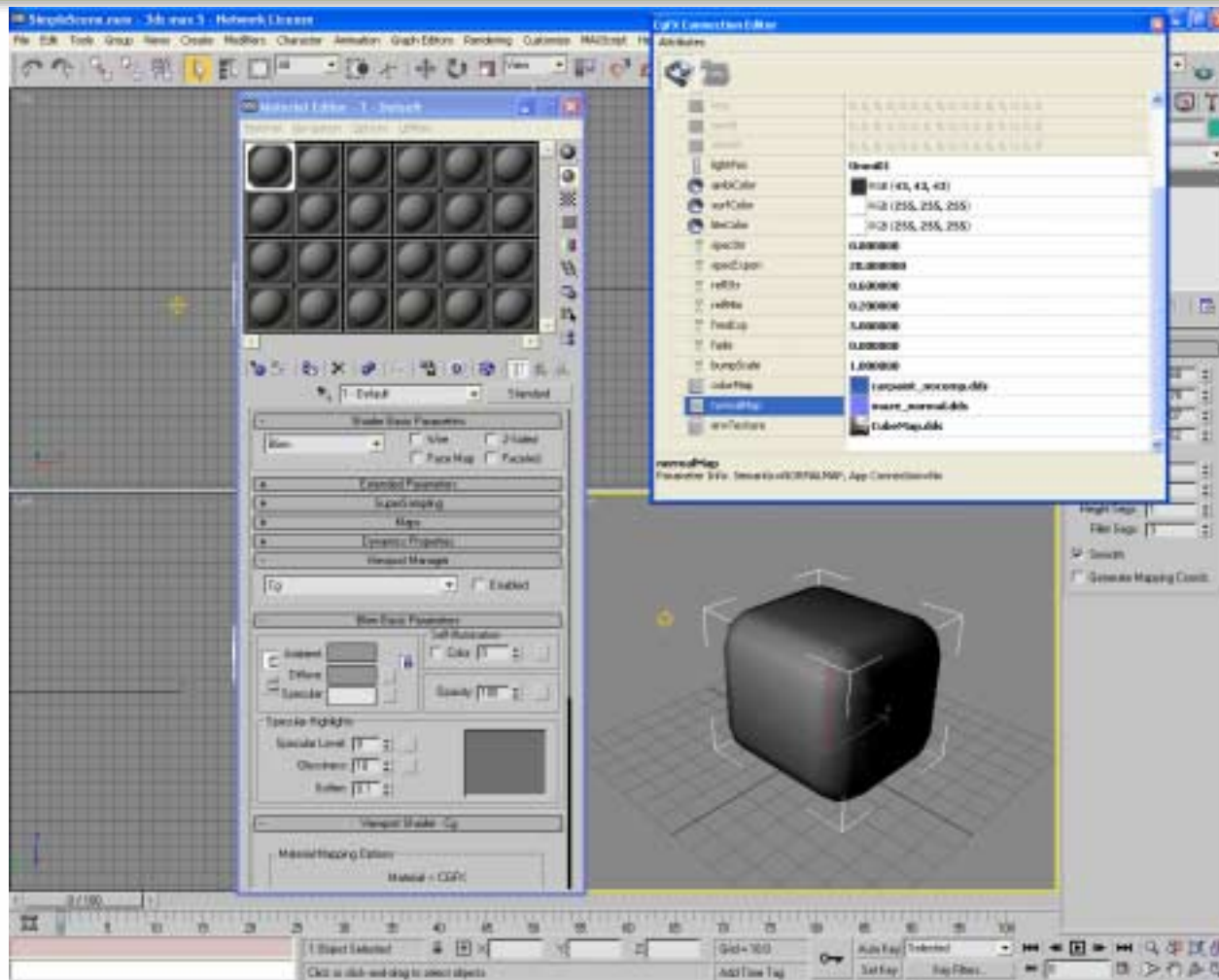
- Connection editor window appears (with “default.fx” as the shader)

Click FX Name – Buttons Appear



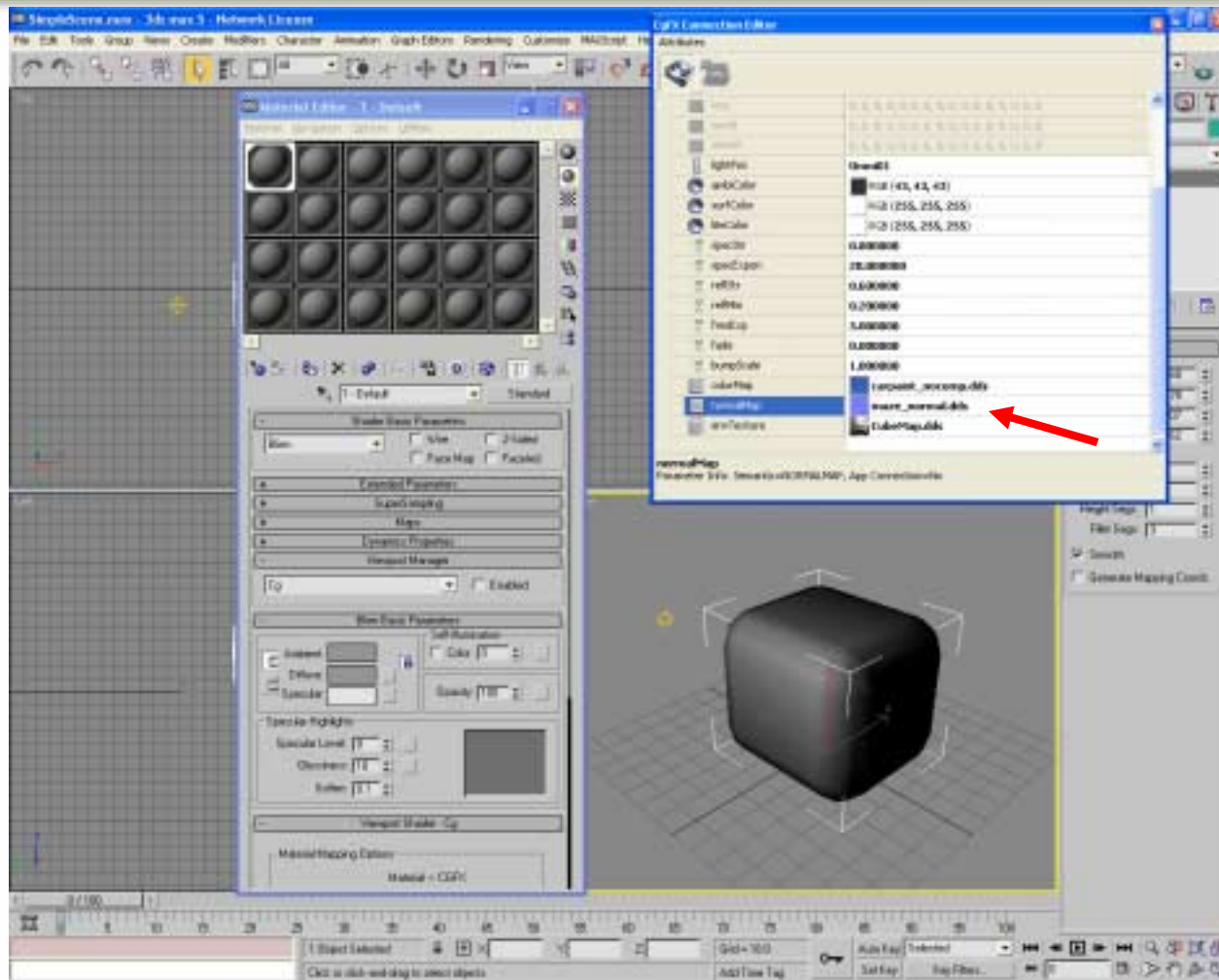
- Click “File” to access different .fx shaders

Shuffle Windows Around So We Can See



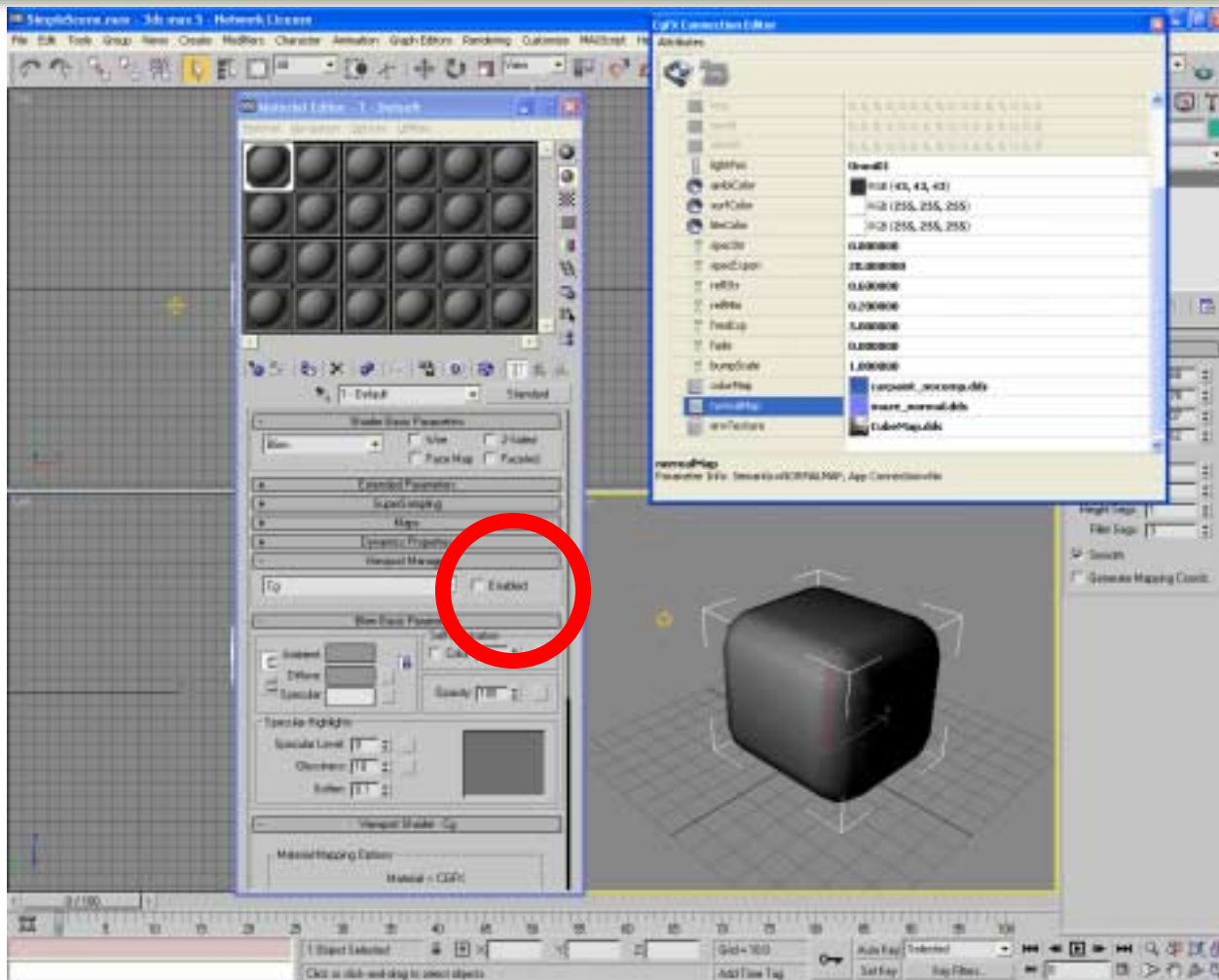
- We want to be able to see the connection editor attributes, material editor, and our selected object.

Set the BumpMap to our “buckles.dds”



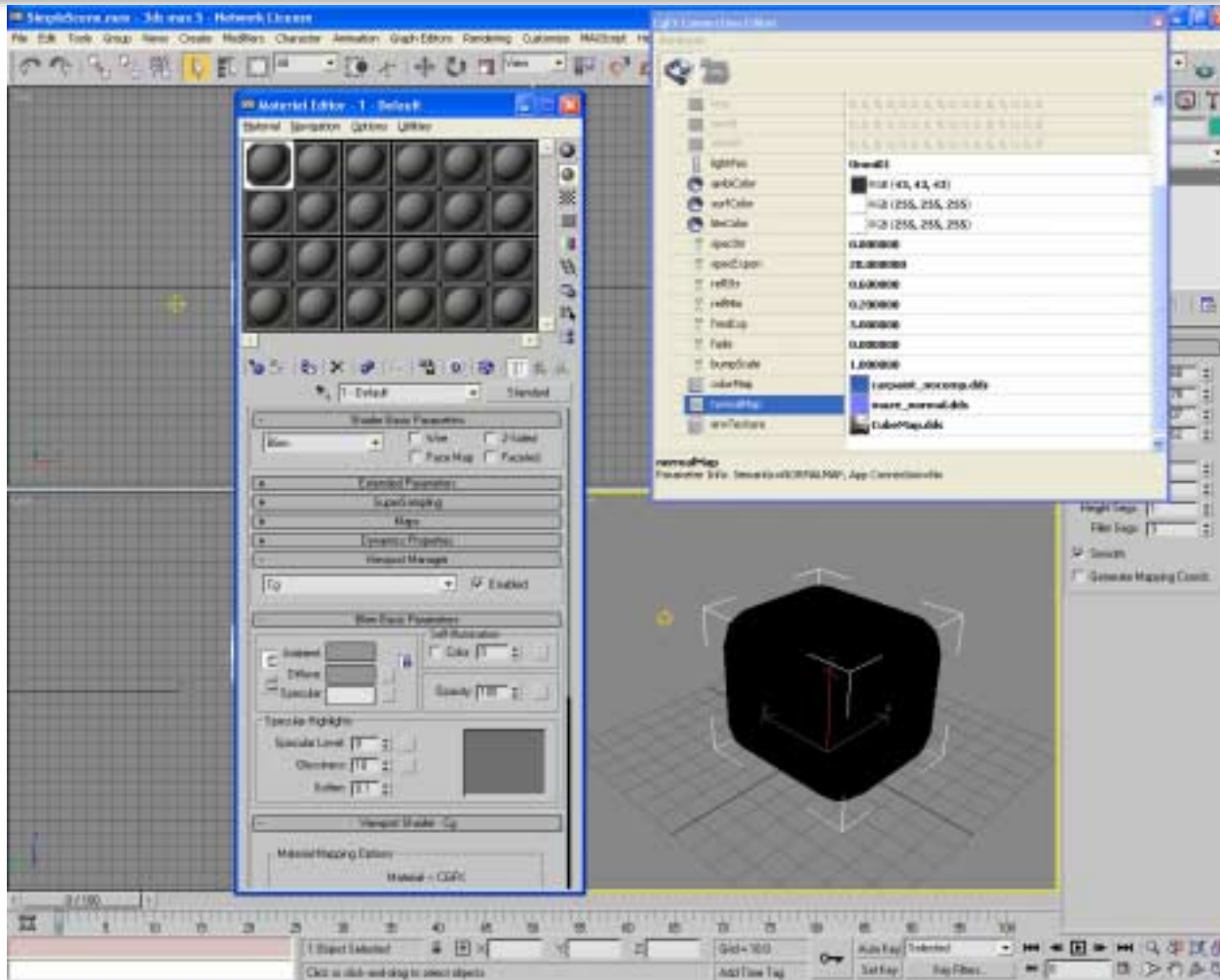
- Click texture name to see “file” button as before

Ready to Enable Cg Viewport Manager



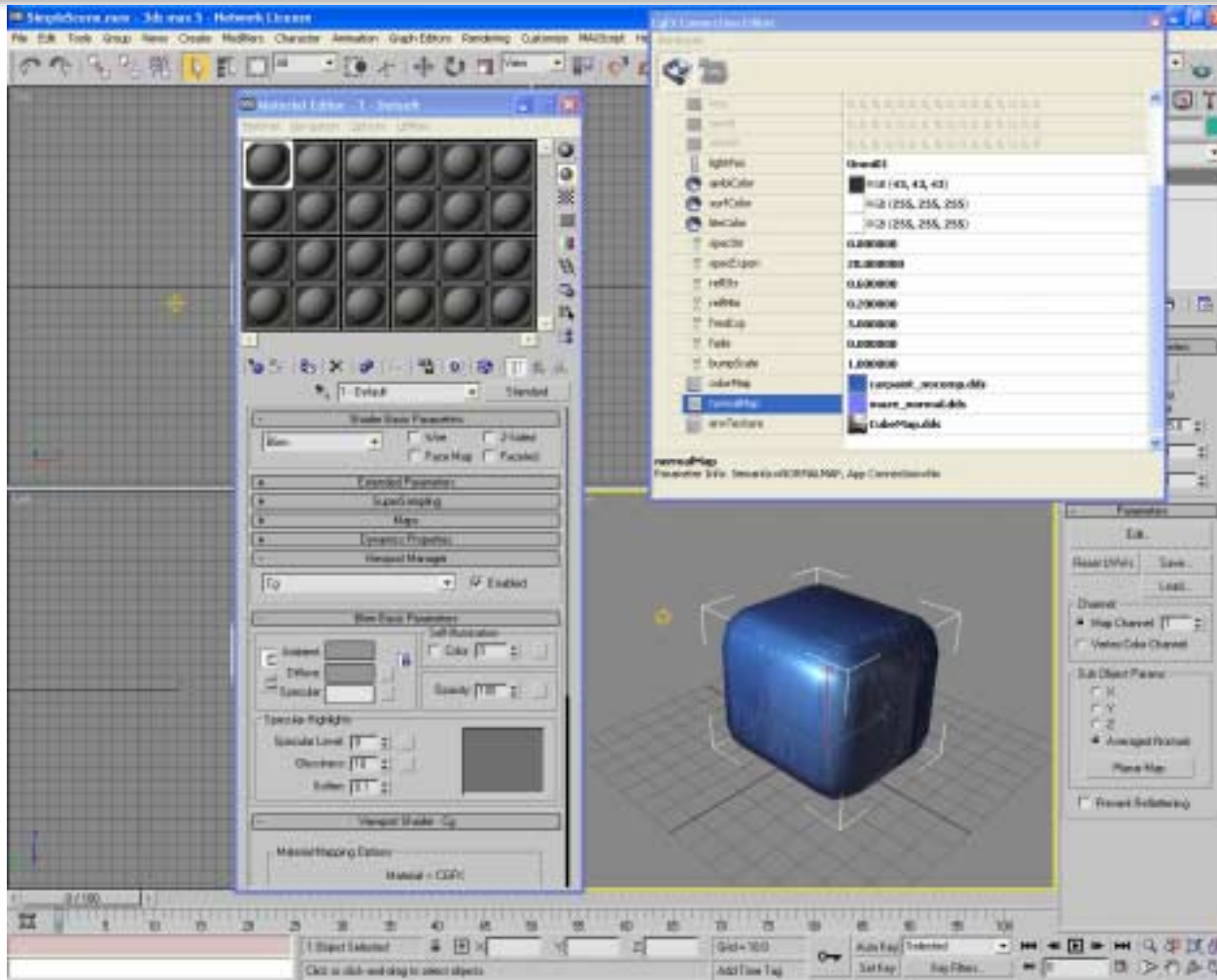
- This is a per-material flag, to help you manage scene complexity

Cg Enabled – Hey, It's *BLACK!*

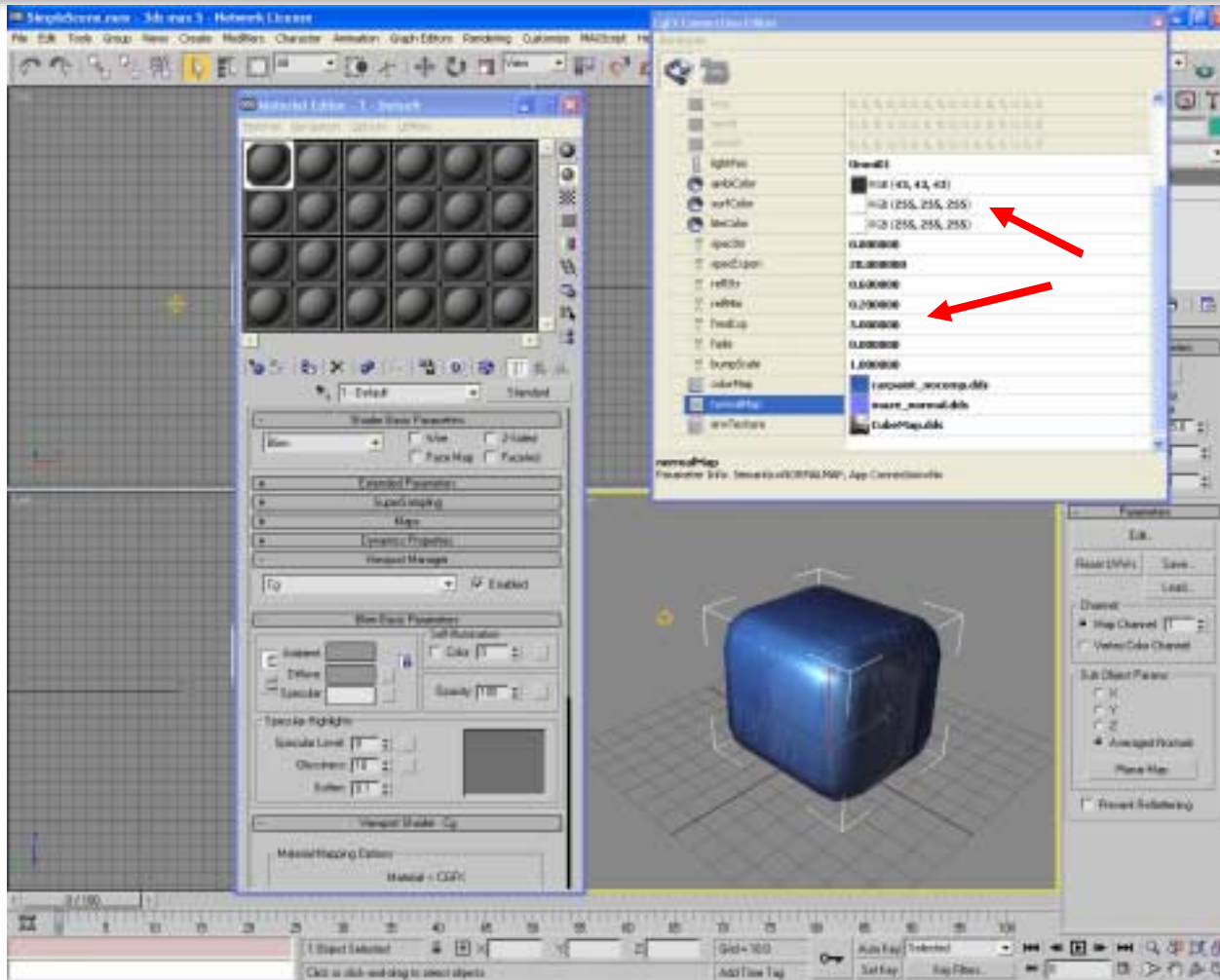


○ What's missing?

Voila!



Tweak CgFX sliders as you like



- Try different techniques
 - “ps11” just shows unshaded paint color
- Change amount of “Bump”
- “Fade” does nothing!

Now We Will Alter The Shader

- Click “Edit” – Starts WordPad (or whatever editor the registry says)

- Comment-out this line (ctrl-F “Fade” match case will jump to it):

```
// float4 paint = tex2D(PaintTex,STcoords);
```

- & Un-comment the next line:

```
float4 paint =  
    lerp(PaintTex,STcoords), (1.0).xxxx,Fade);
```

- ...then Save (ctrl-S) and Exit the Text Editor
- Now the “Fade” slider will actually do something

Load Our Changes into Max

- Looks good!
- Sliding “Fade” will desaturate the paint color

See Shader on a Large, Complex Model



Playback
Slider

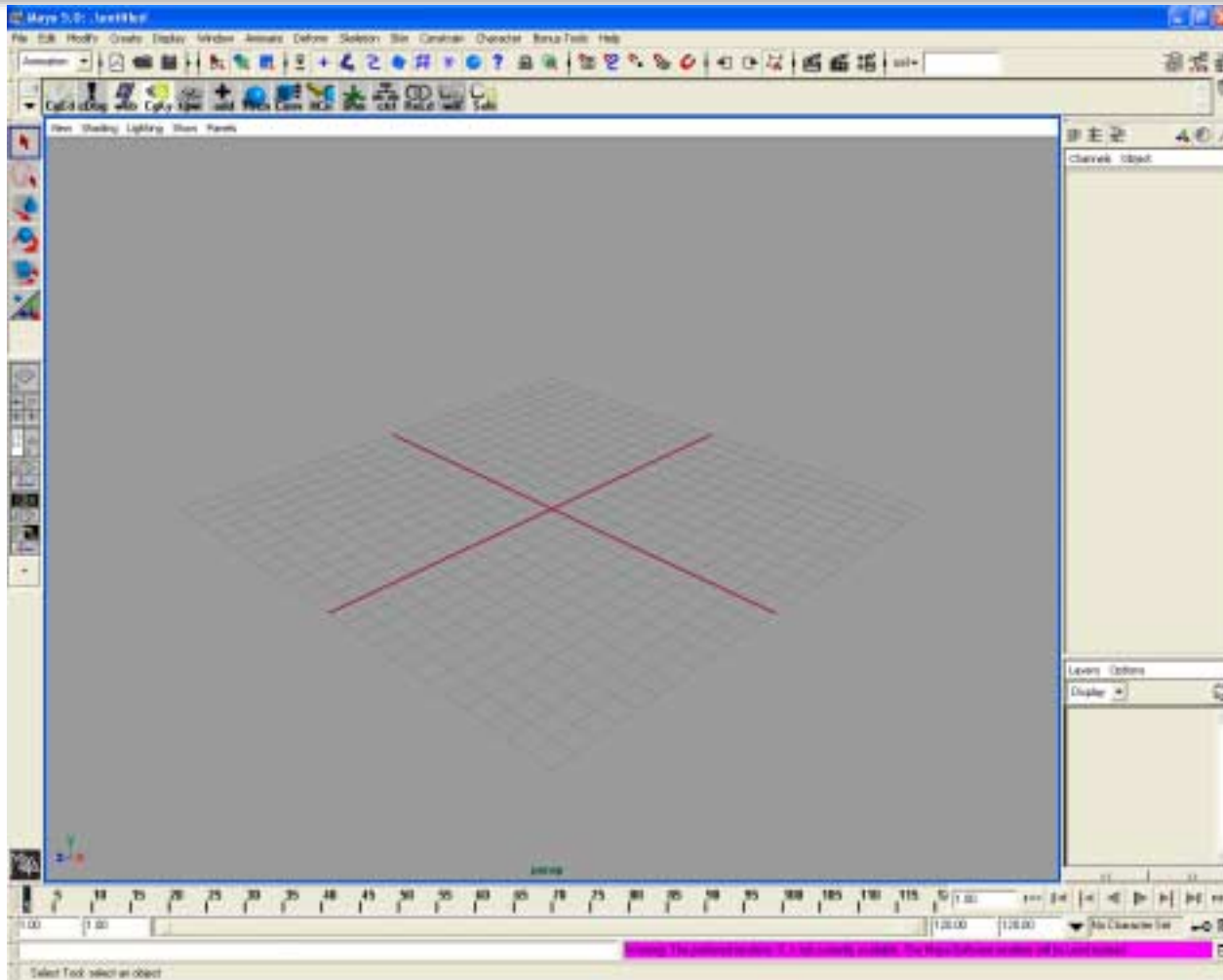
- Scene “truck-Siggraph.max”
- Slow update, but that’s from the *heavy* model
- Final demo can be realtime even with such a complex model

Start Maya

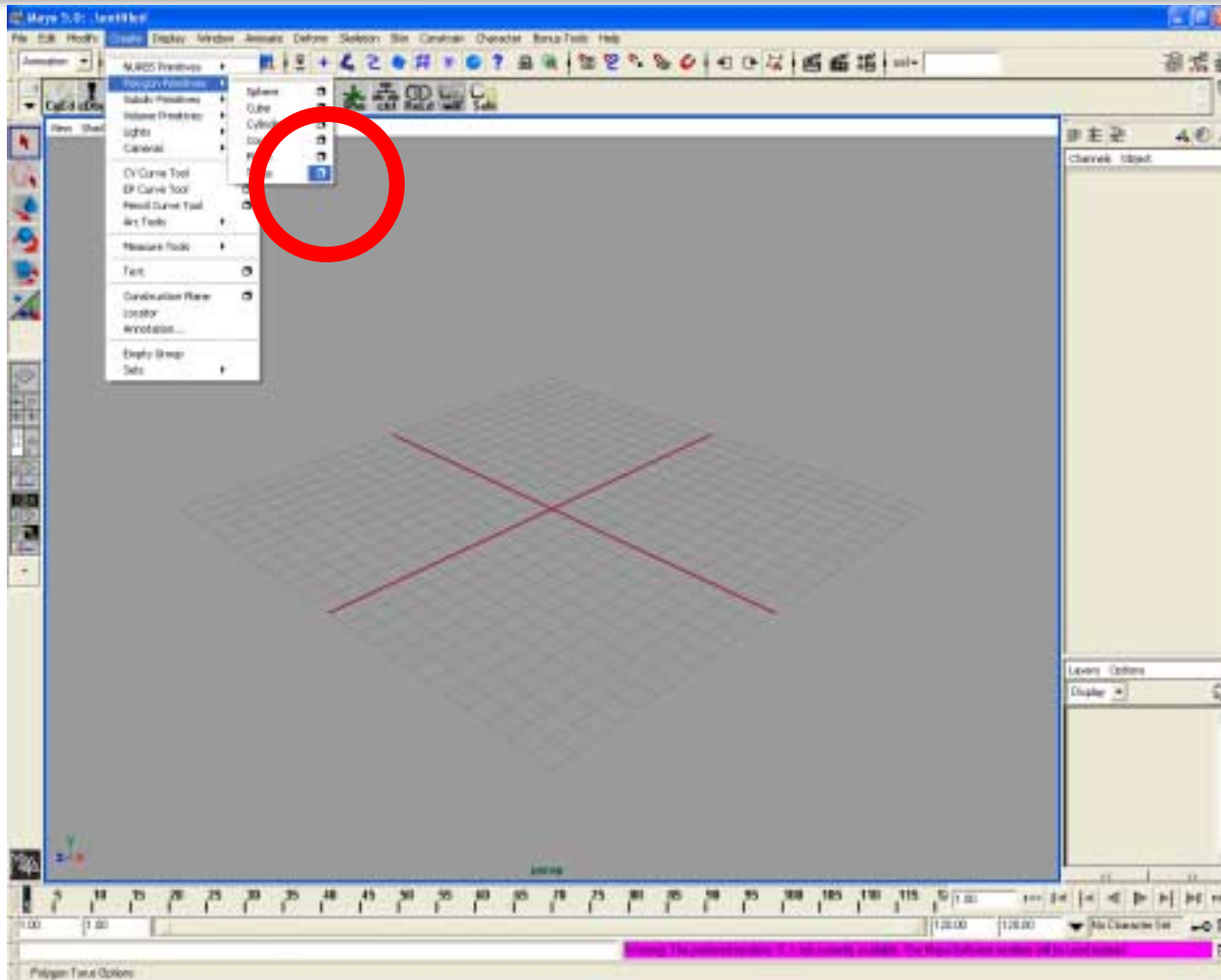
- Create Poly Torus (load demon?)
- Radius 8, Section Radius 5, twist 0, subdiv axis ~36, subdiv height ~20, Y axis TEXTURE CHECKED
- Right Button OVER TORUS to create shader node
- Click file folder to load our same shader
- Technique “ps2”
- click middle mouse on view
- press “6” – press “Sele”
- Create->Lights->Point Light
- Press “w”- drag yellow center square away from torus
- click once on torus
- click “Sele”
- drag to “pointLight1”
- **now light is connected to our same shader**



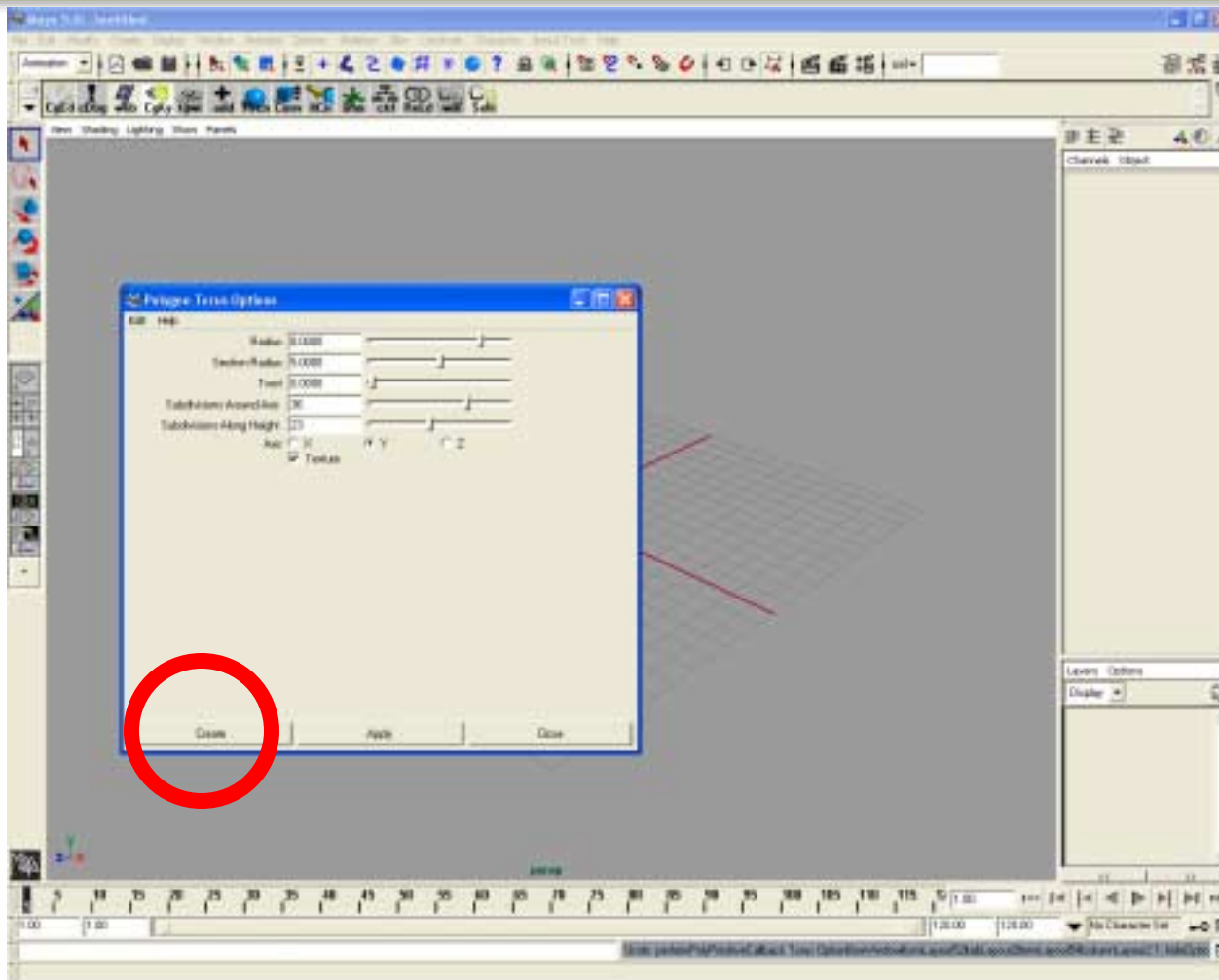
Start Maya



Create Poly Torus

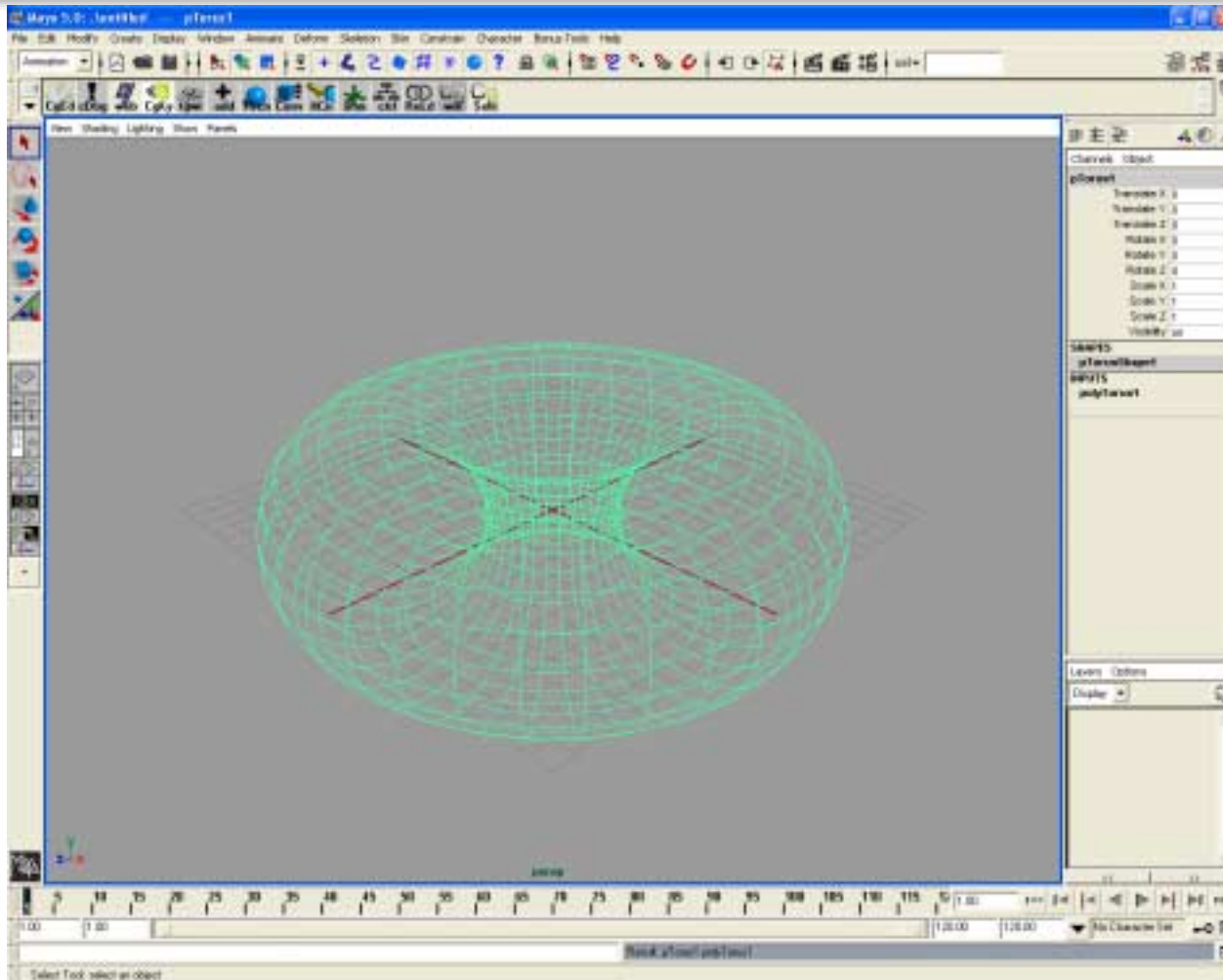


Options

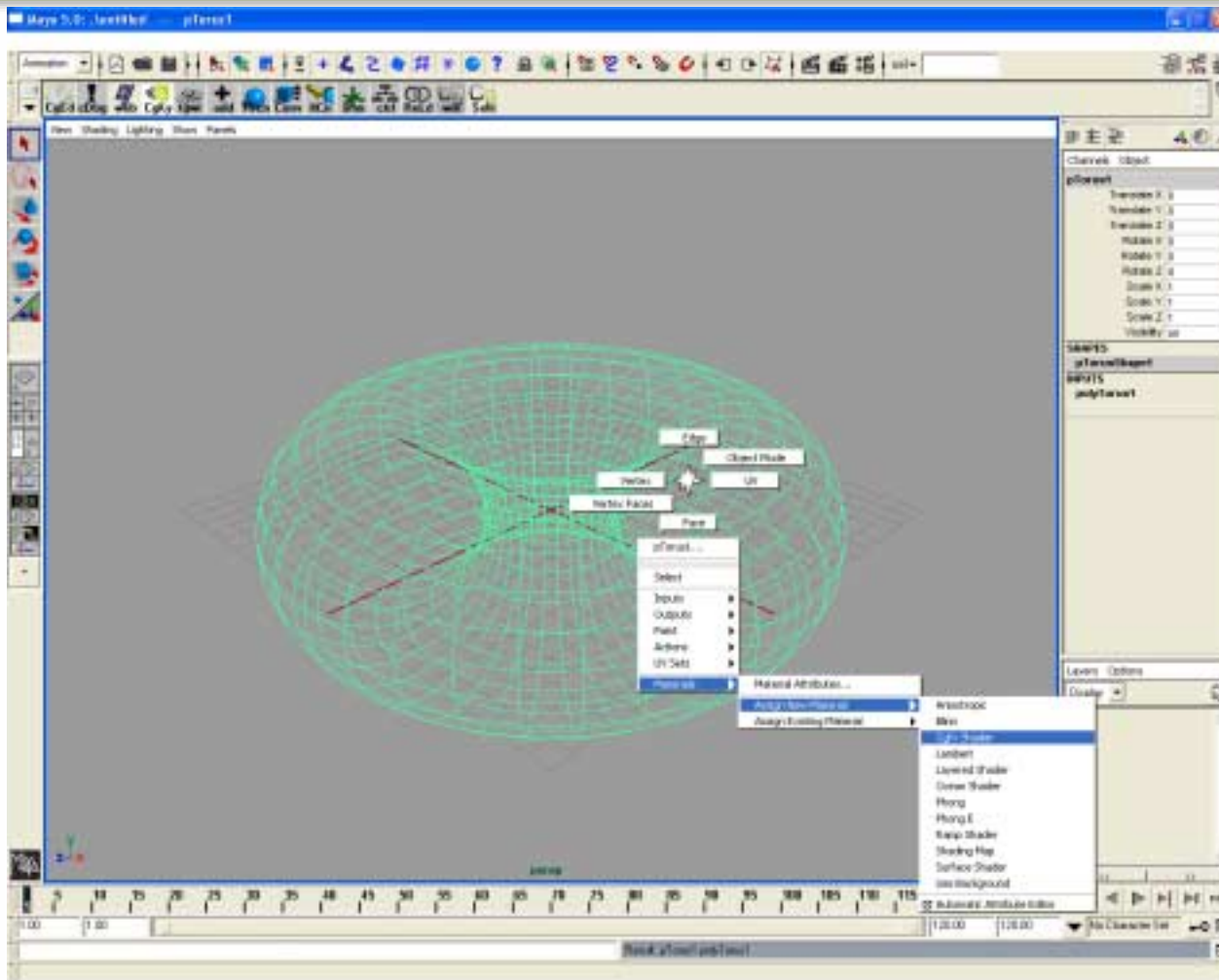


Radius 8
Section
Radius 5
twist 0
subdiv axis
~36
subdiv height
~20
Y axis
TEXTURE
CHECKED

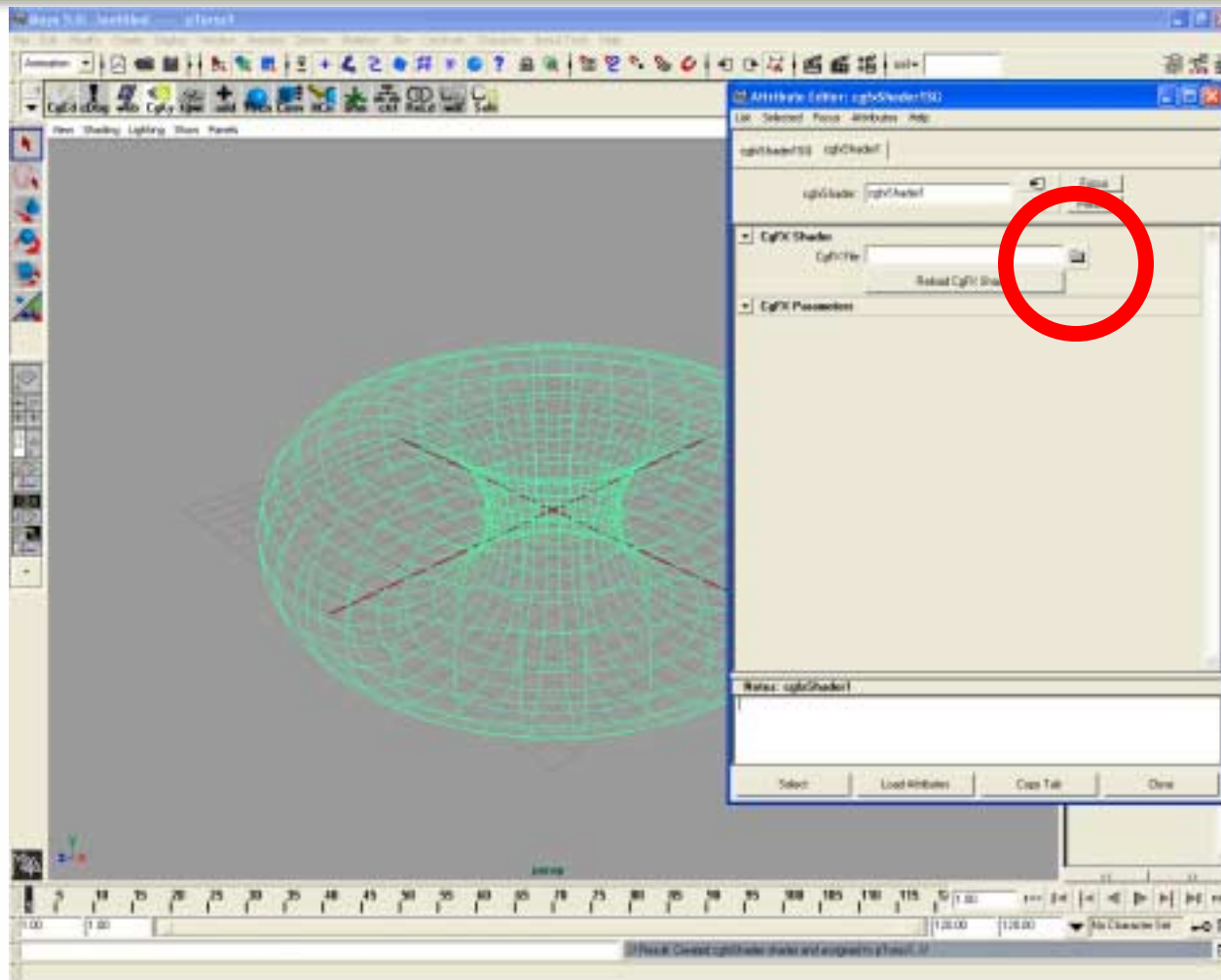
Wire Torus



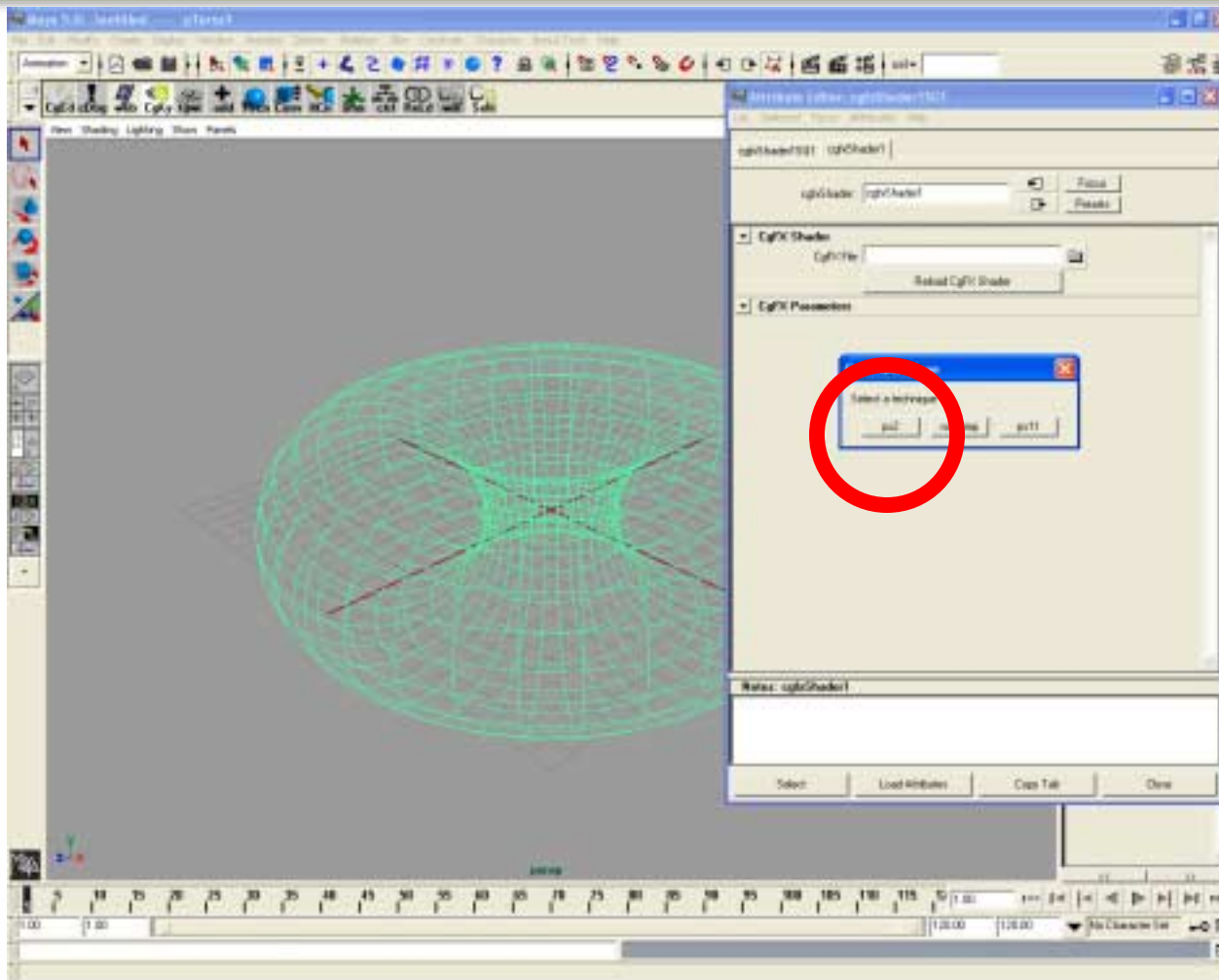
Right Mouse Select Over The New Object



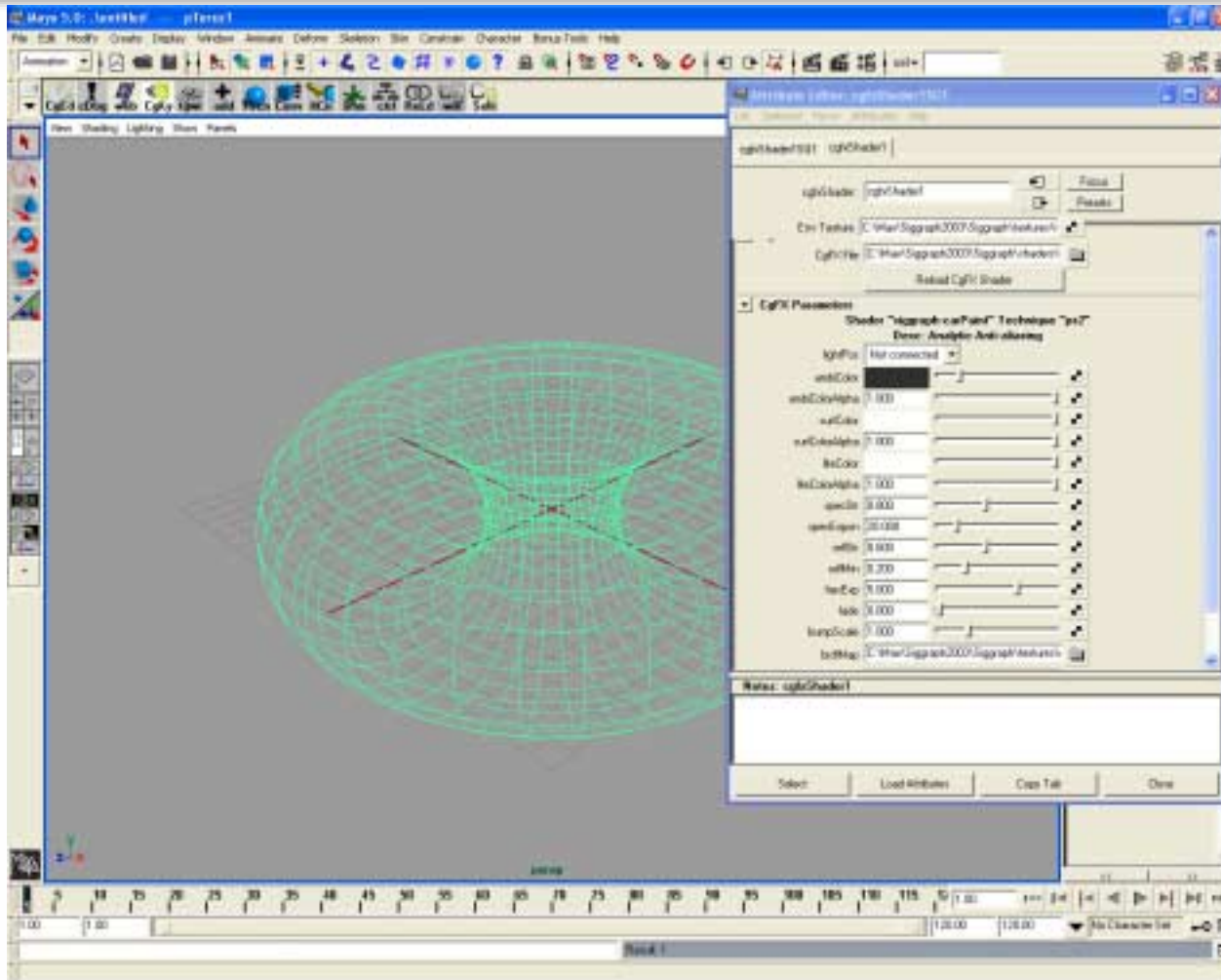
Attribute Editor Will Appear



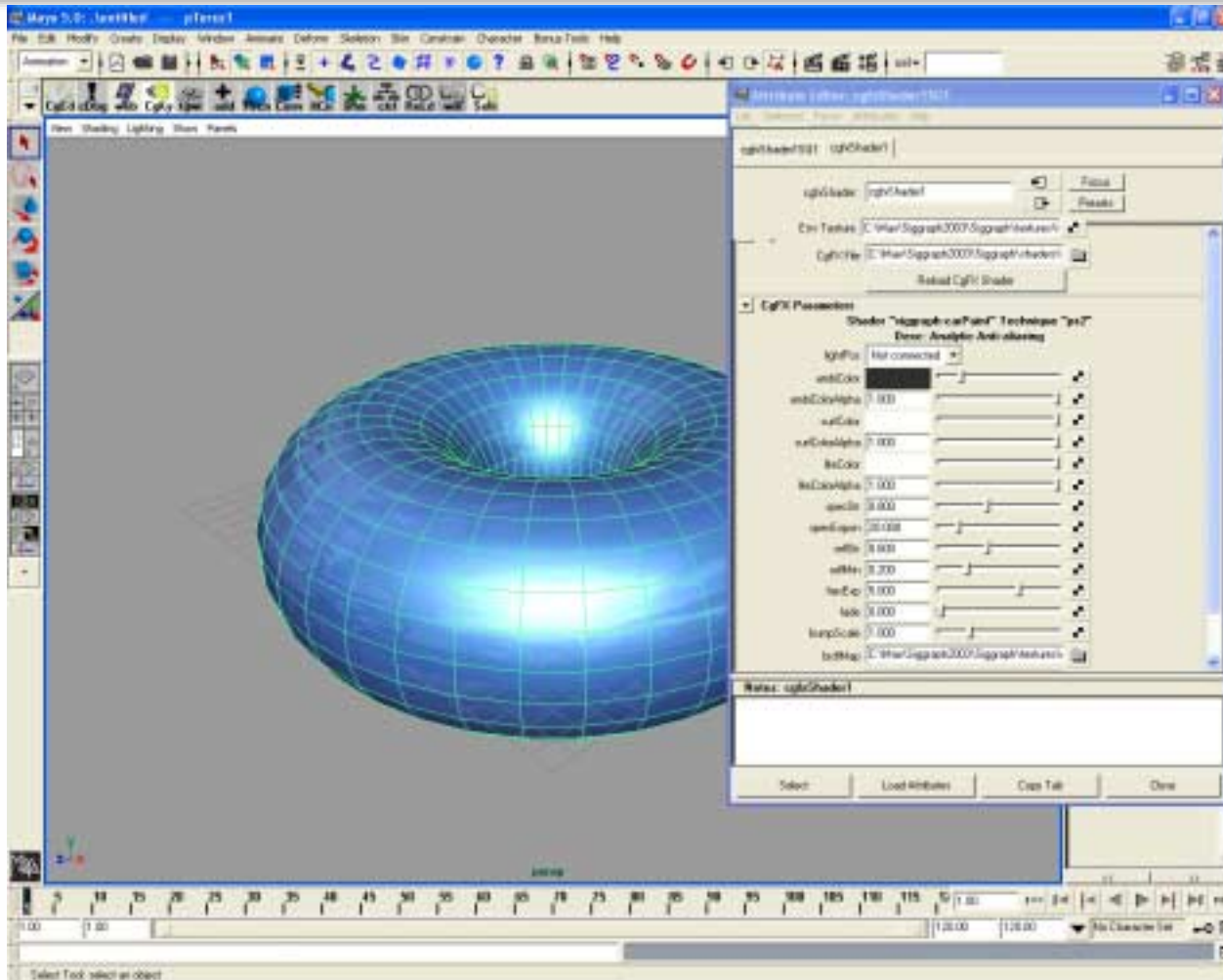
Select – Technique Choice



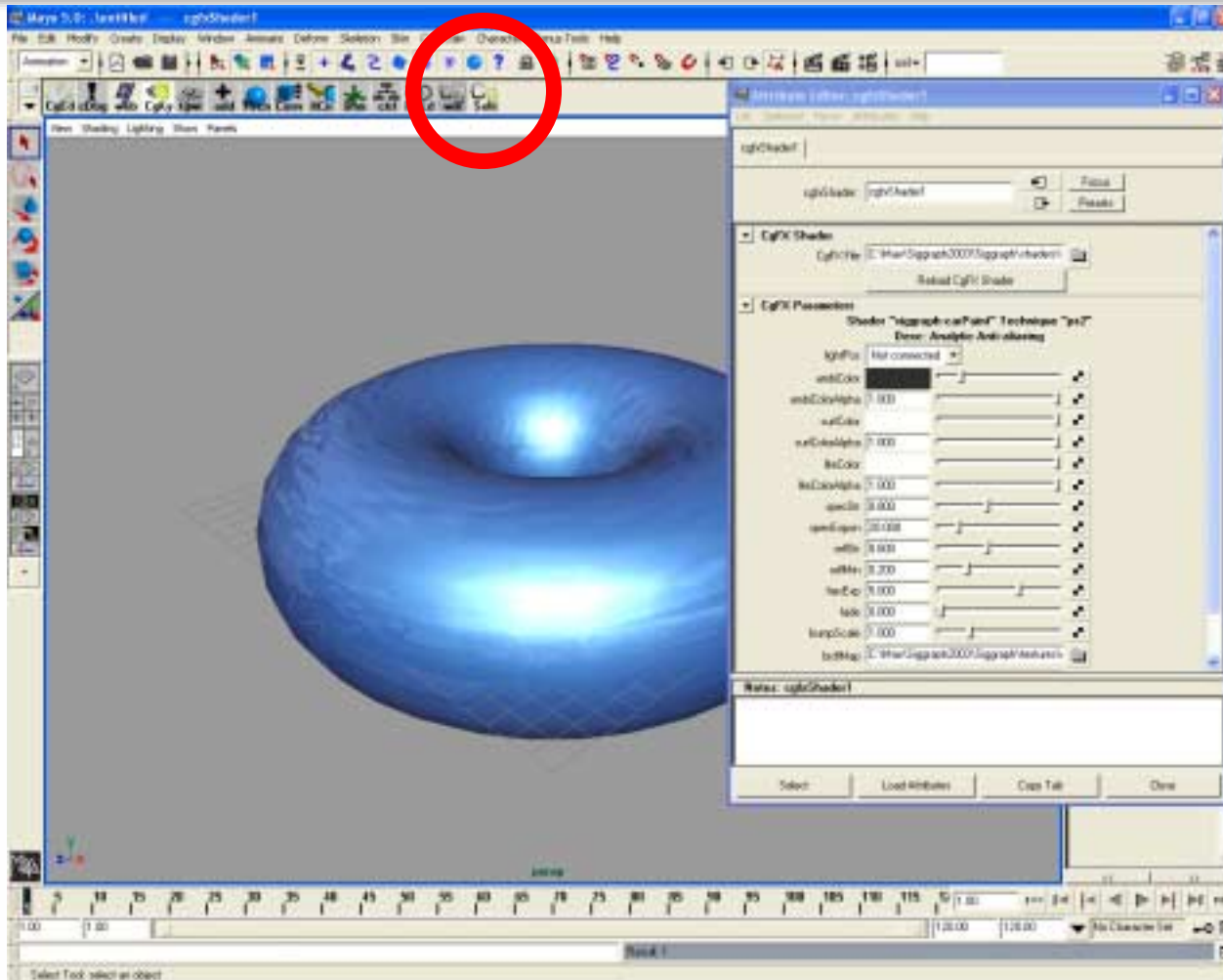
Attributes Appear



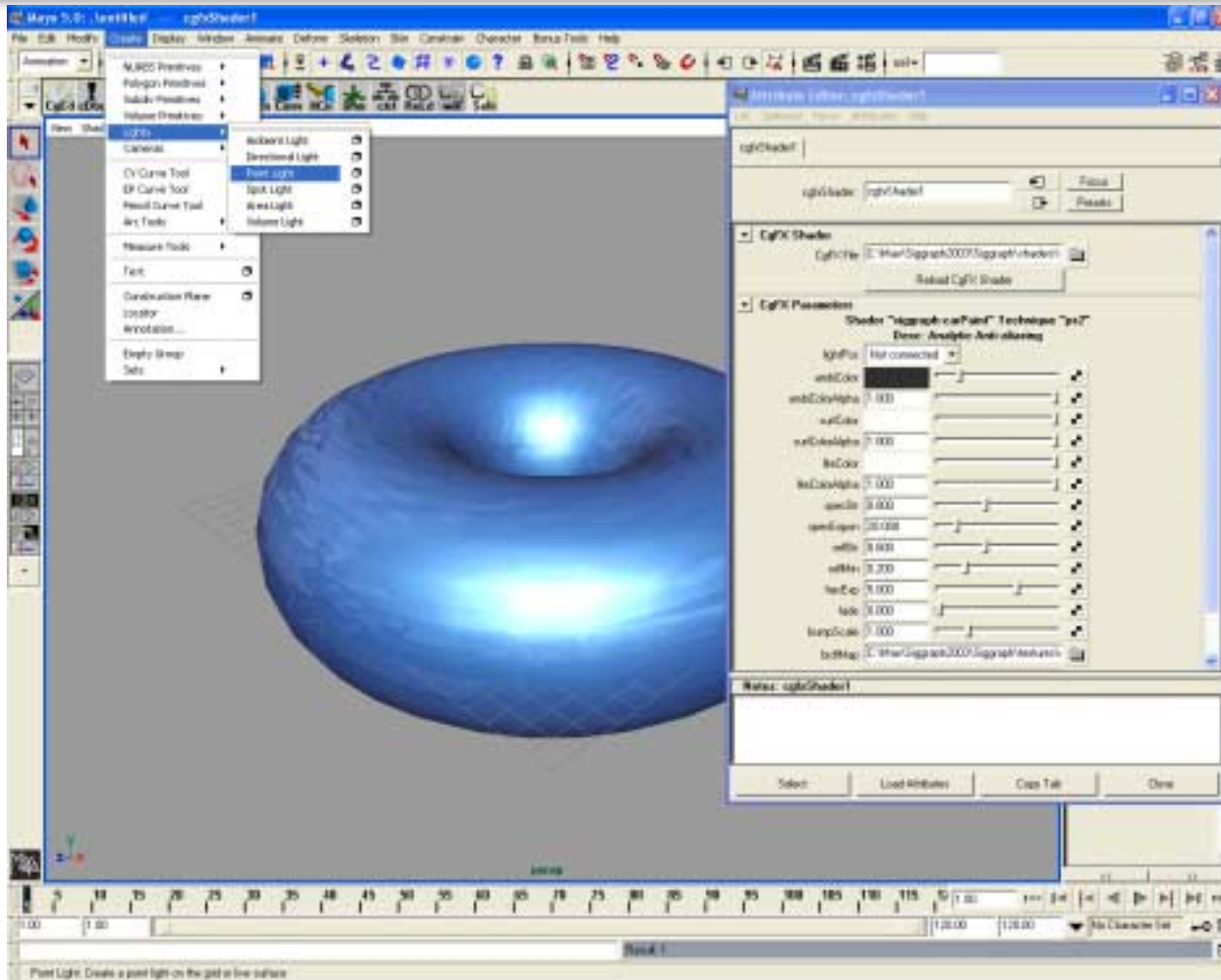
Select Win w/Middle Mouse, and press “6”



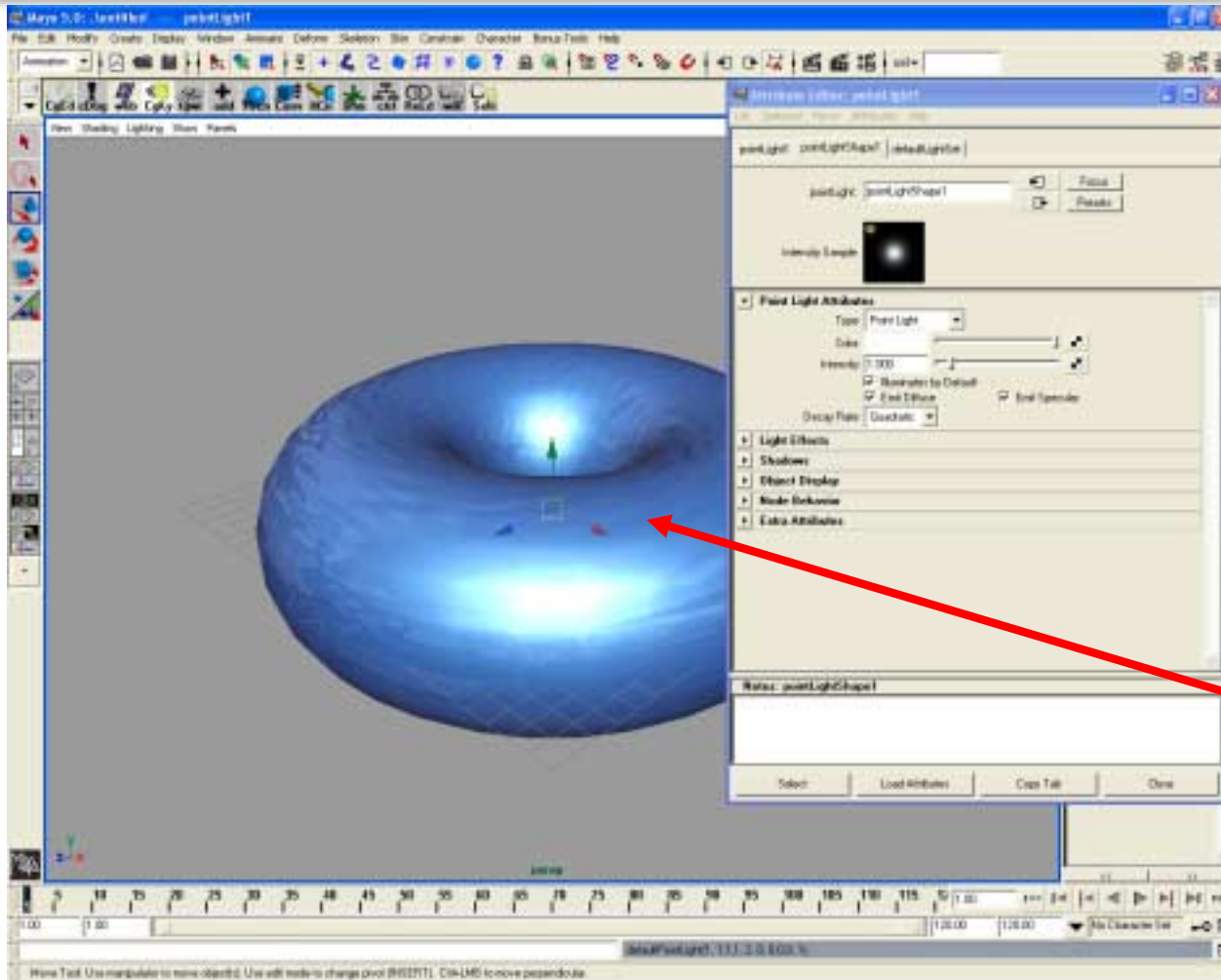
Press “Sele” button to shift selection



Create->Light->Pointlight Menu

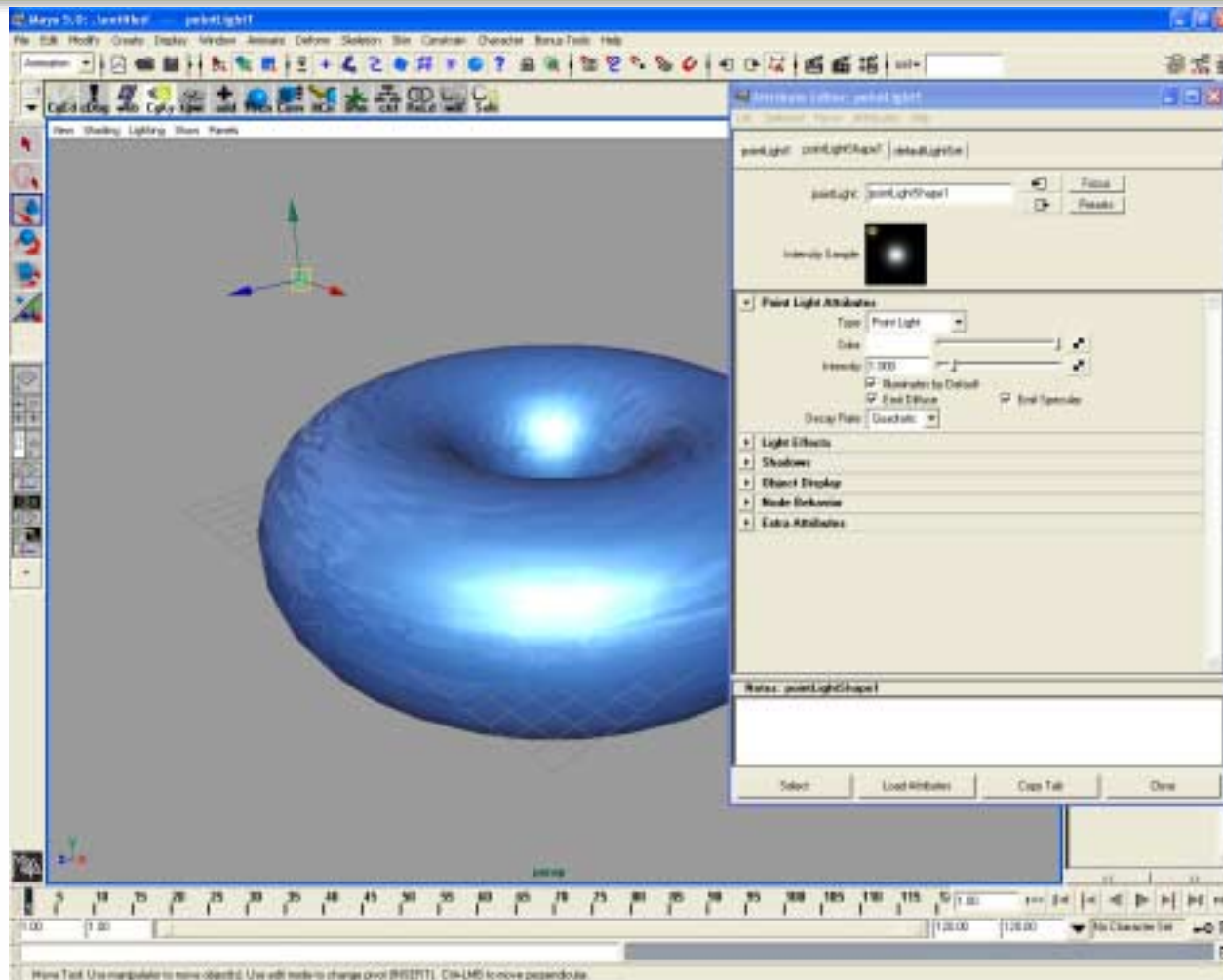


Press “w” to see manipulator

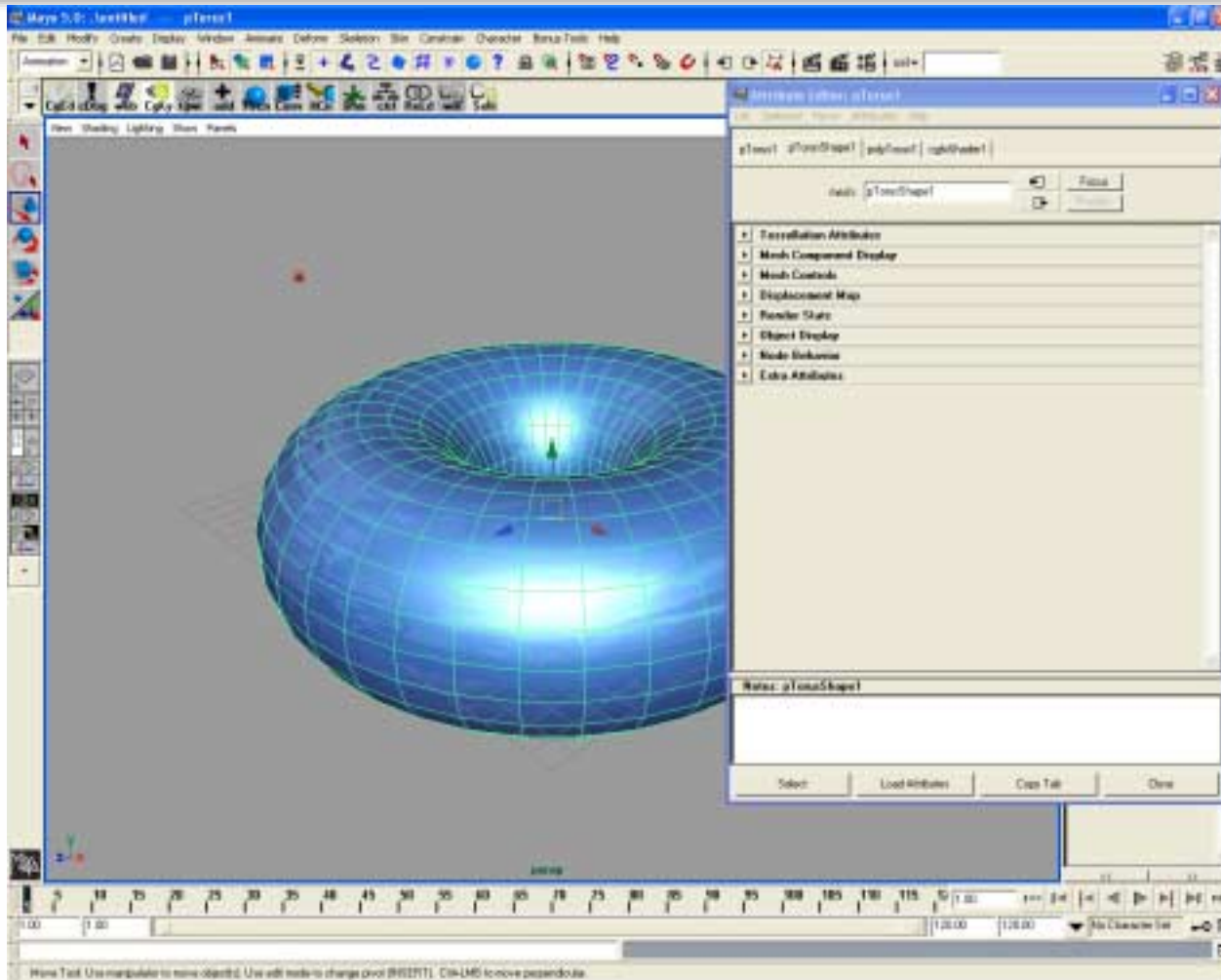


Manip at Origin

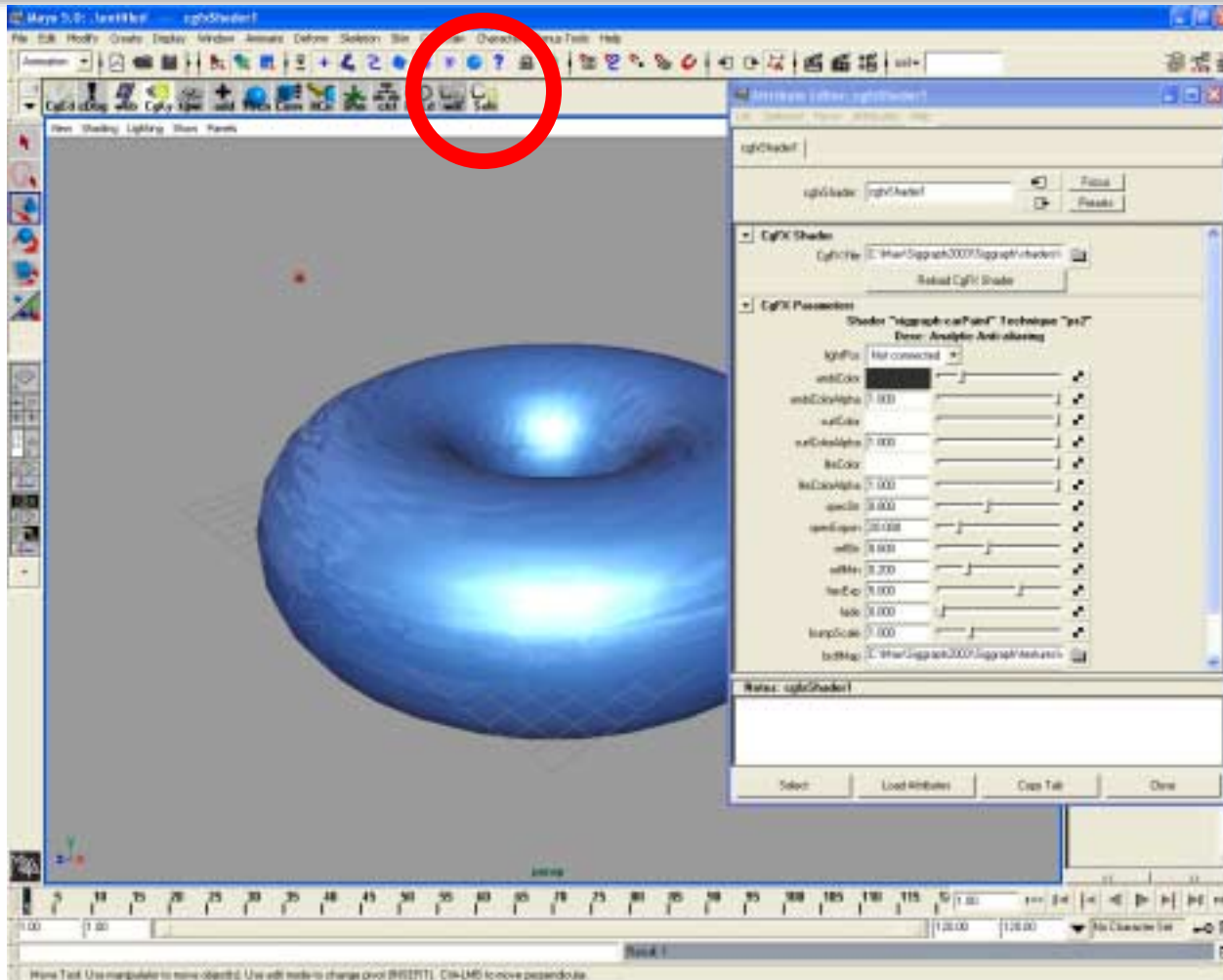
Drag Yellow Square to Move Light



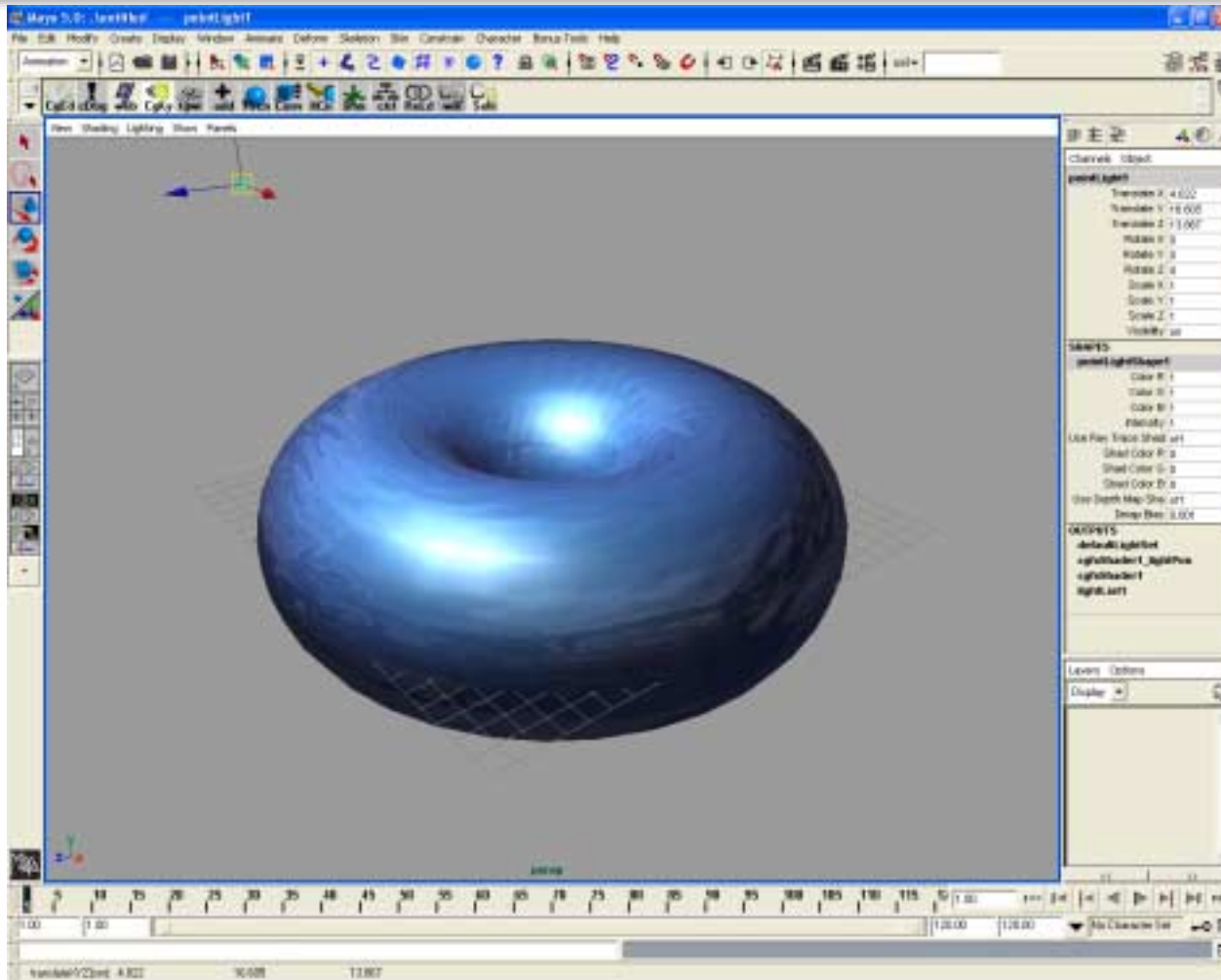
Click Torus to Re-Select It



Click “Sele” to Select Shader



Lighting is Connected!



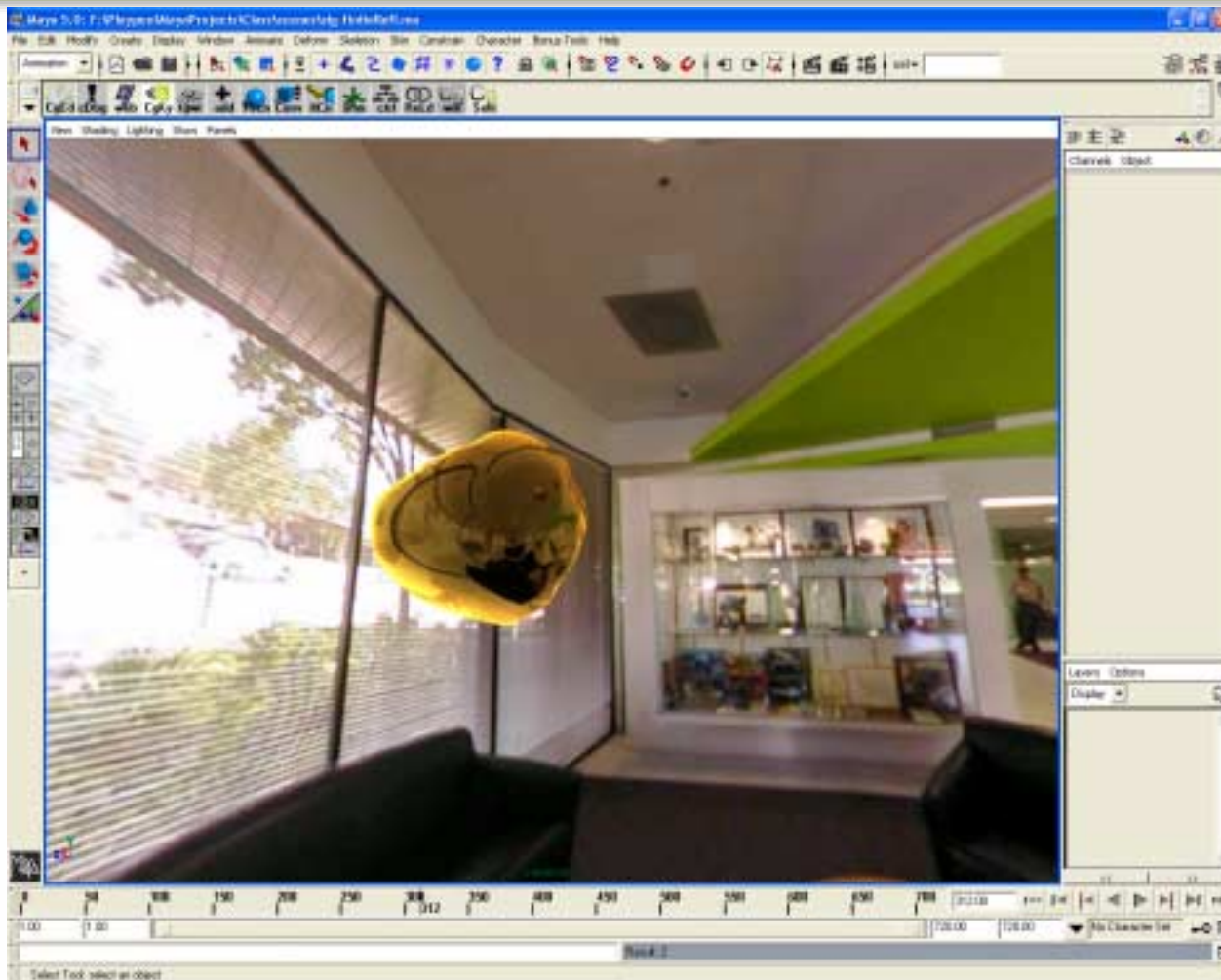
- Drag light around to see its effects
- Move camera by holding down “Alt” while dragging

User-Defined Coordinate Systems

- In this example, the coordinates of the room-shaped cube are used to create “local” reflections on the gold object.
- We pass the room’s World Inverse matrix to the vertex shader, and send the pixel shader a transformed copy of “P” via a TEXCOORD.
- Special Maya node “matrixElements” makes these connections “live” – otherwise we’d use Maya expressions

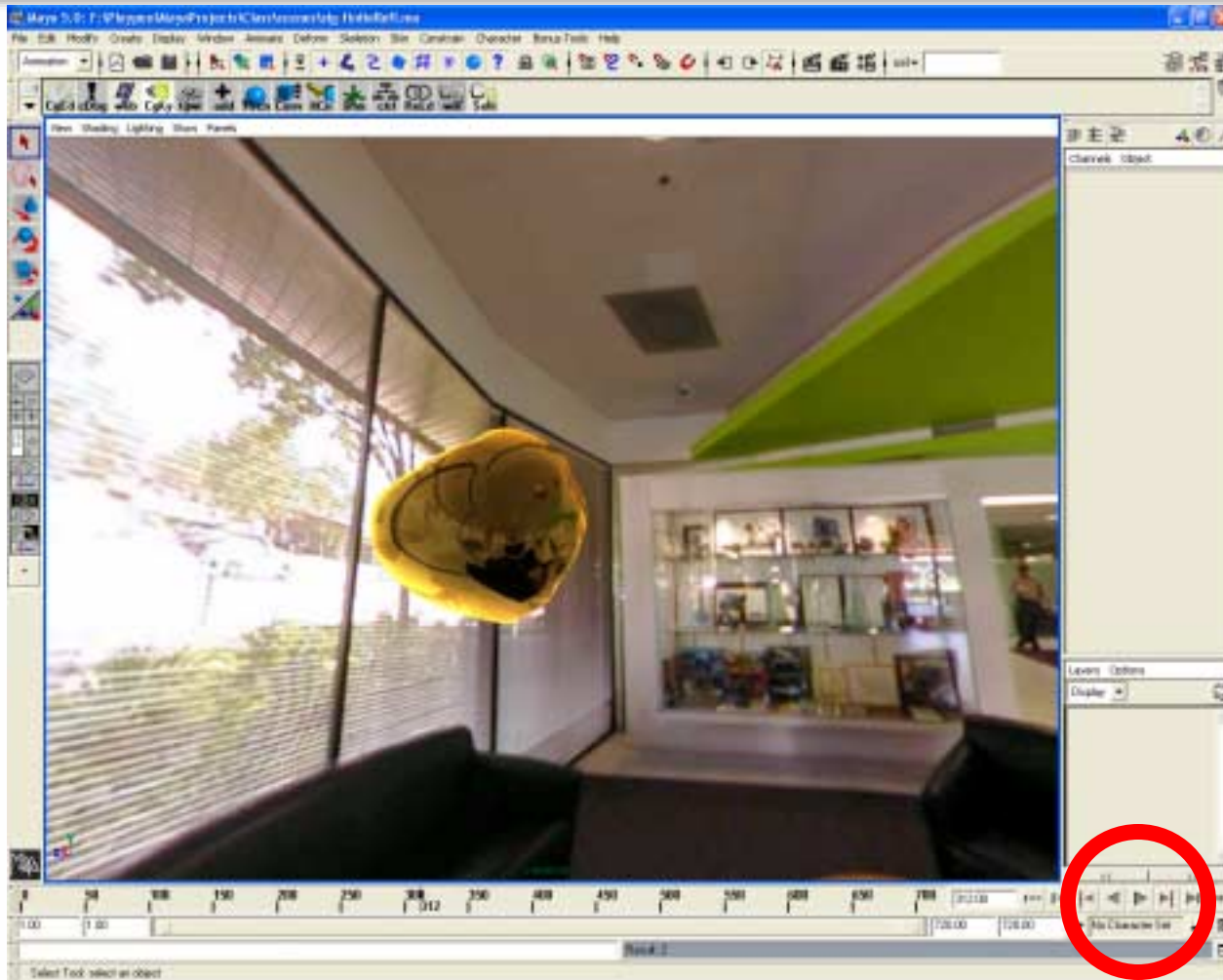


Load “sig-finiteRefl.ma”

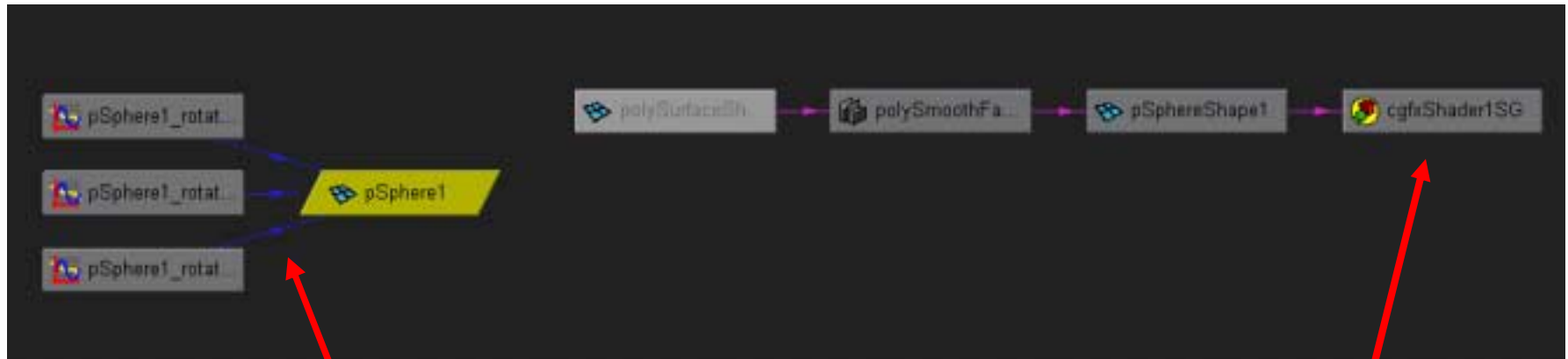


- Scene “sig-finiteRefl.ma”
- Don’t save existing “untitled” scene
- Only two objects:
 - Distorted sphere
 - Cube

Press “Play” to see animation



Hypergraph of “pShere1”

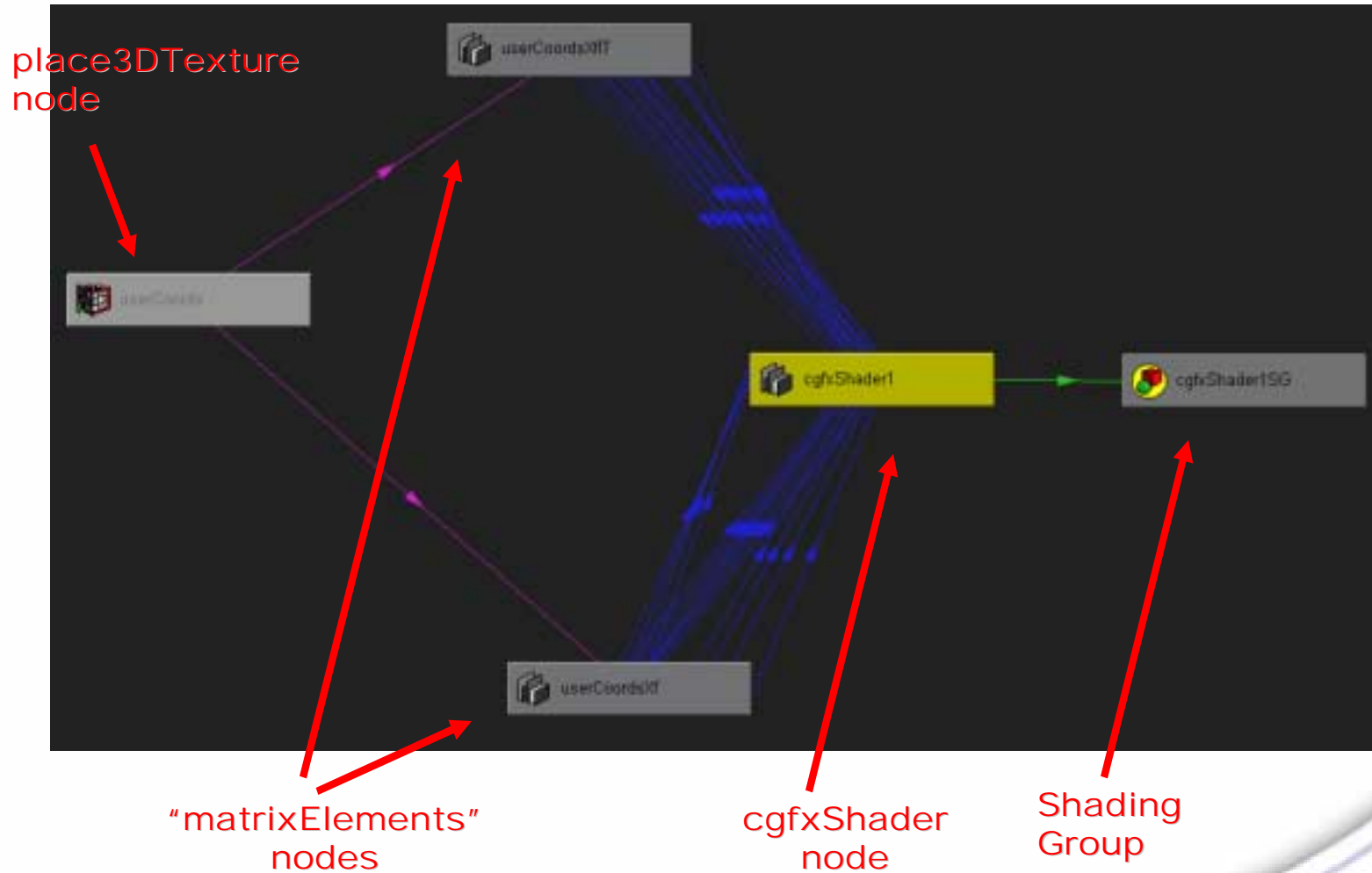


Modelling
Operators



Shading
Group

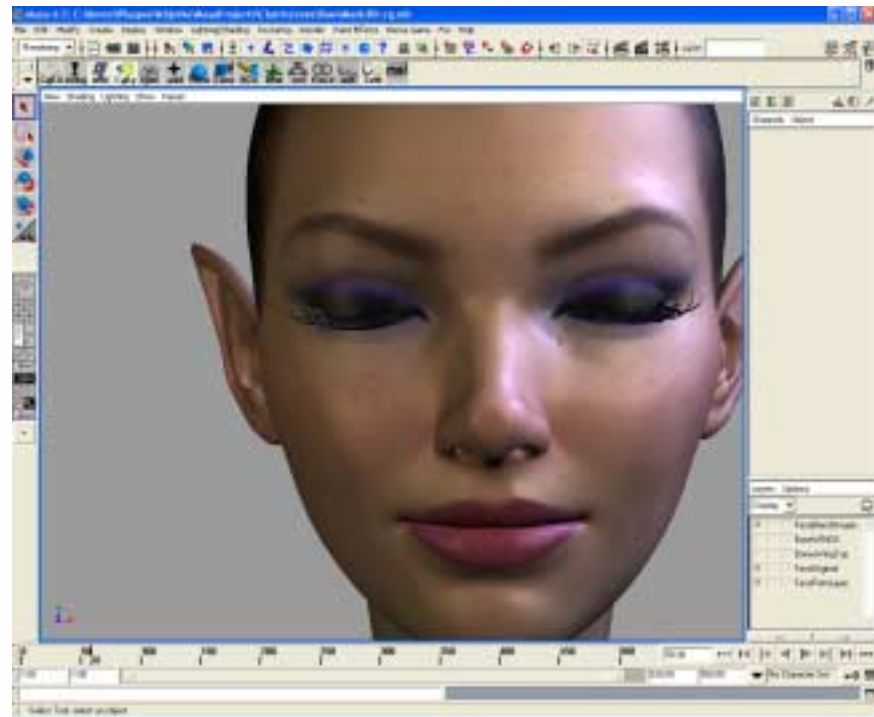
Hypergraph of cgfxShader Node



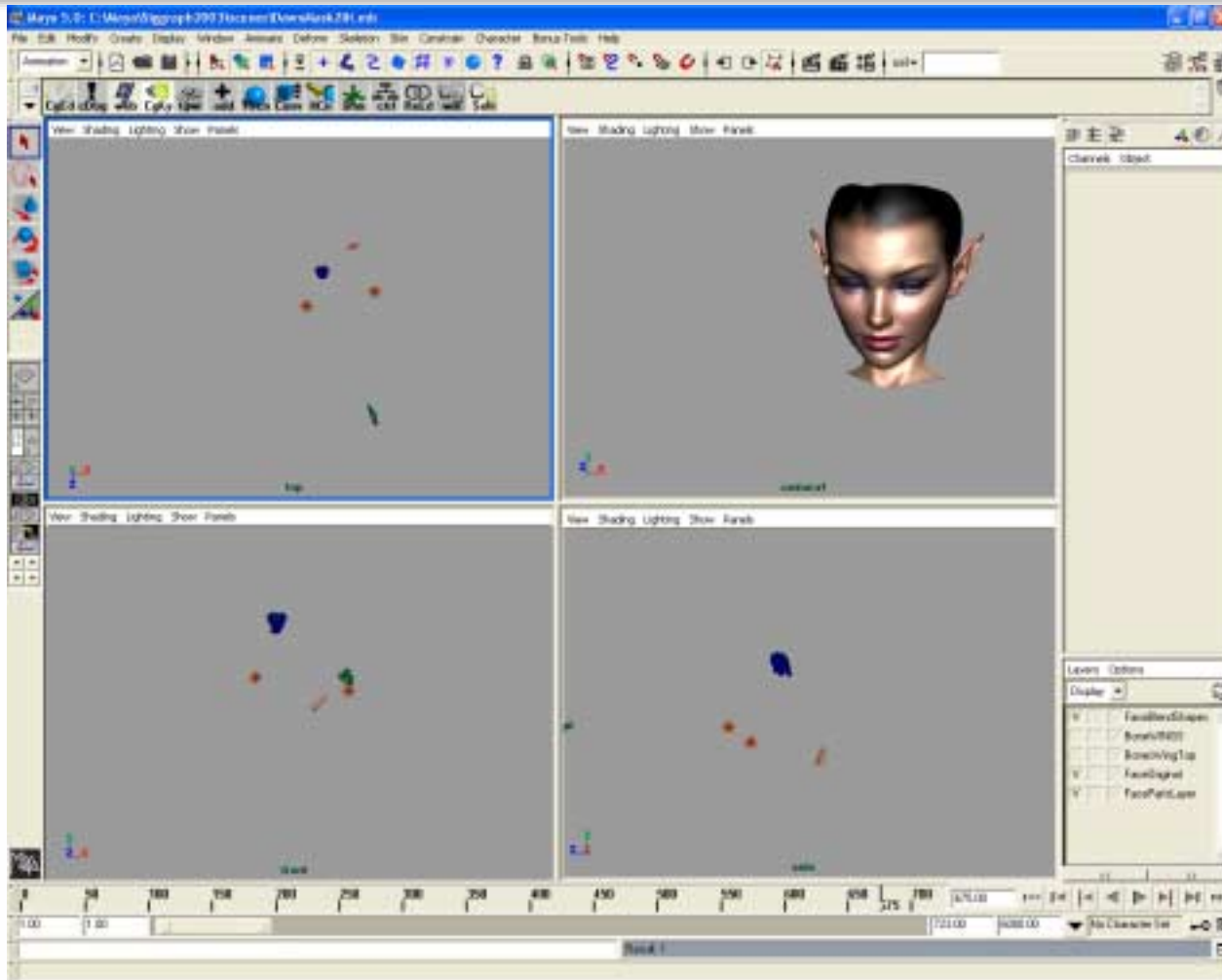
Automated Maya Conversion

- Load scene
DawnMask2lit.mb
- Use “Conv” button
- Answer “Yes”
- Select “nv3x” 4 times
- Hit okay
- Now Dawn is dark

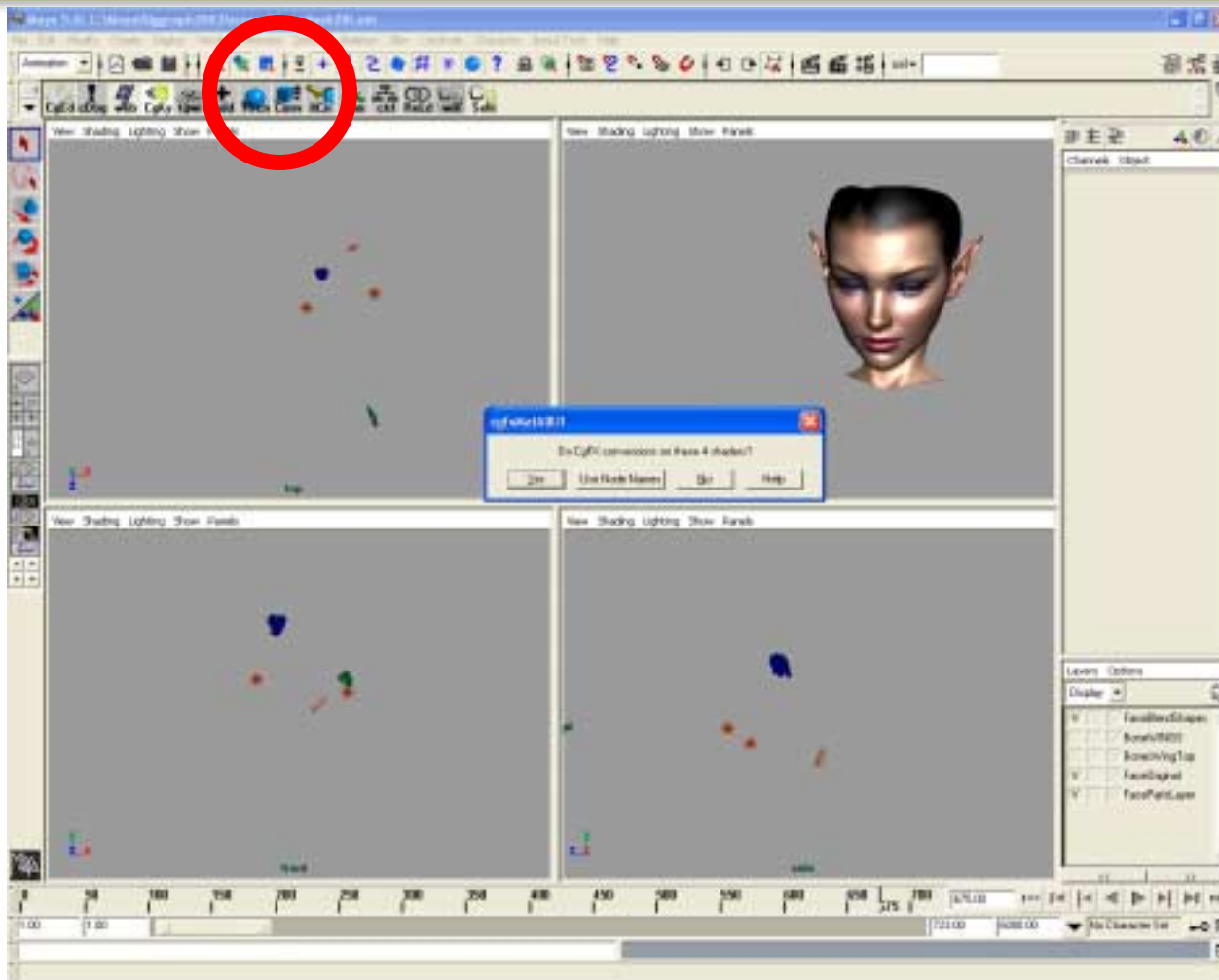
- See how shaders are linked to Maya materials
- Tweak light positions/intensities
 - (farther bright lights shade more evenly)



Load “DawnMask2lit.mb”

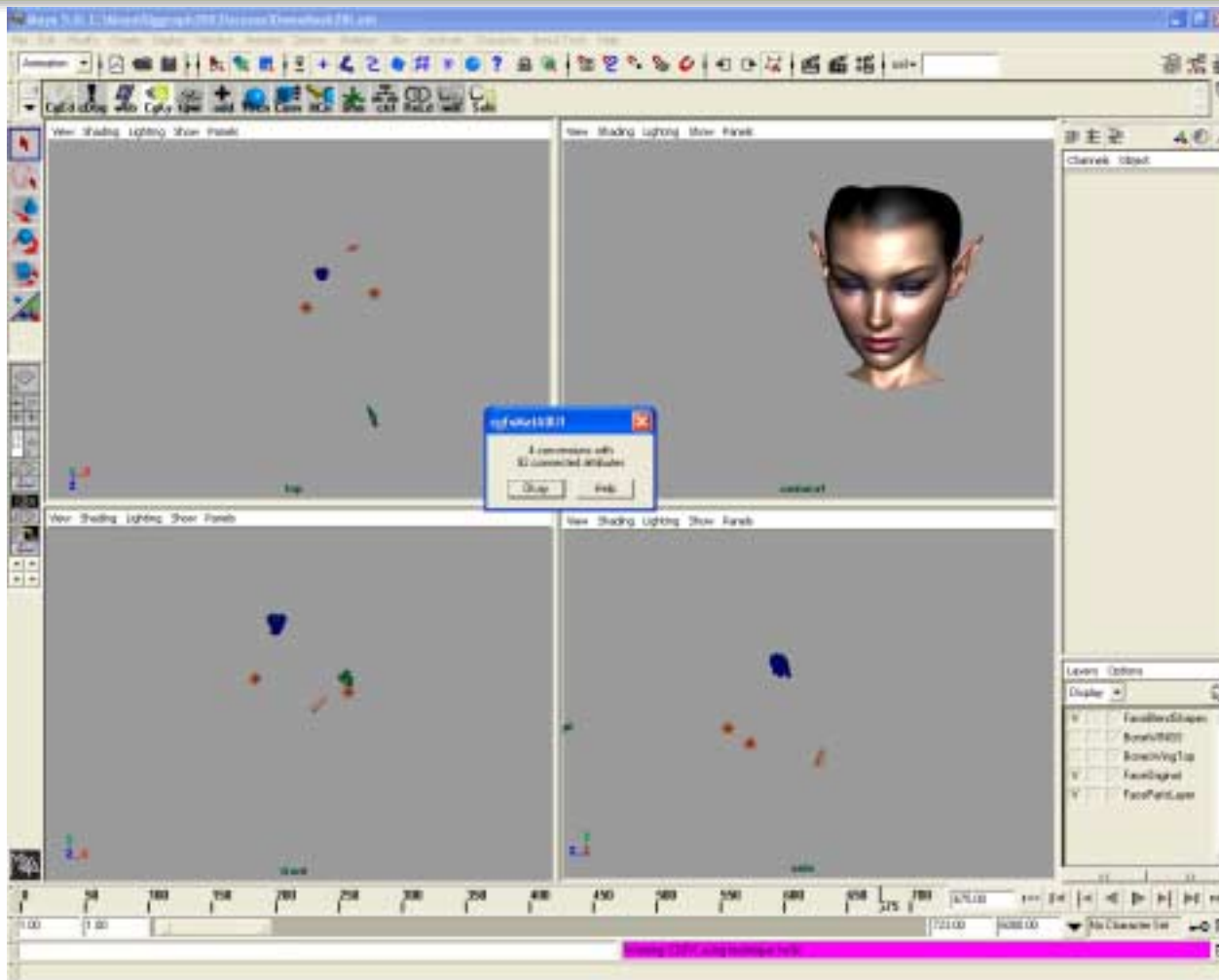


Press “Conv”

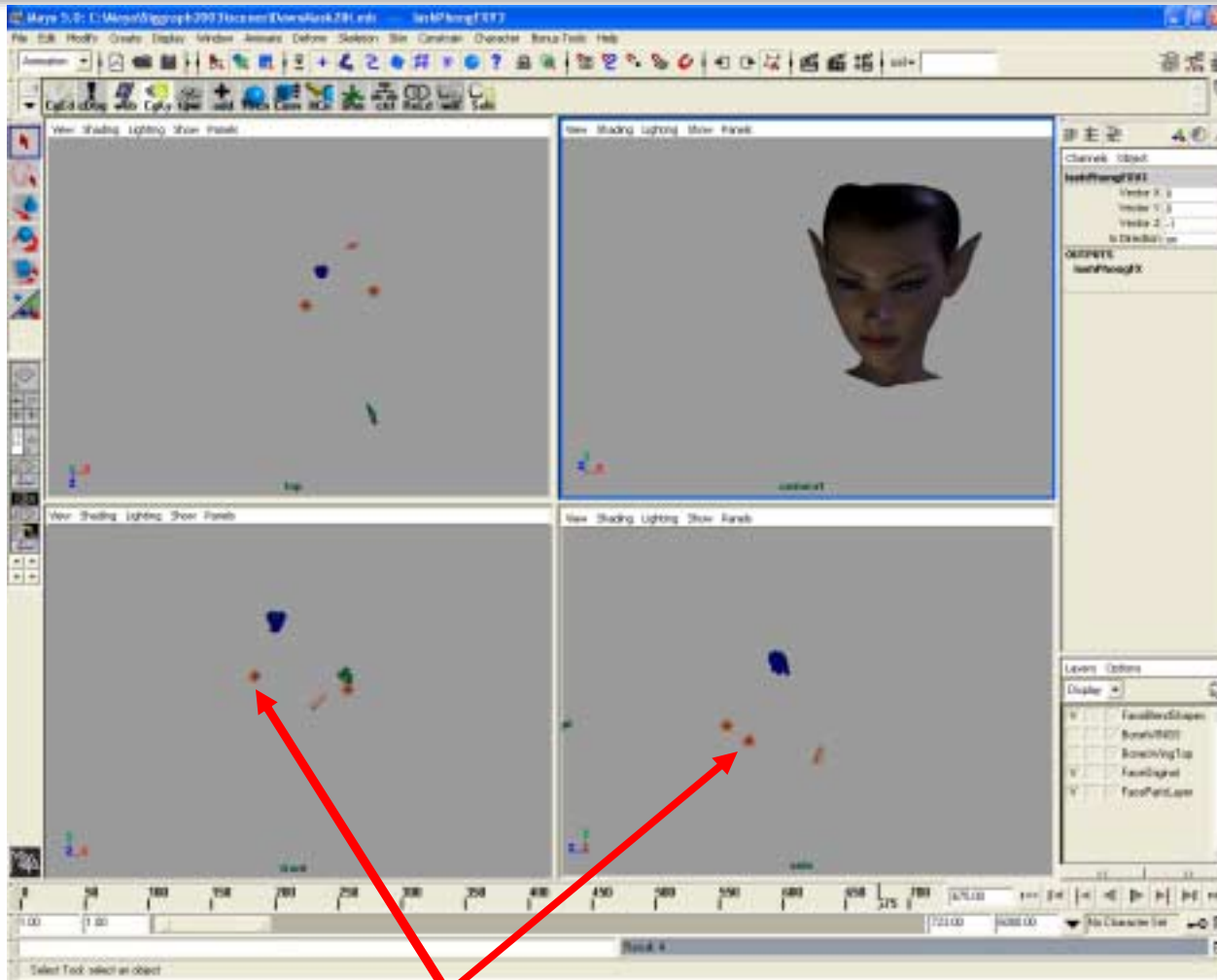


- Answer “Yes”
- Answer “nv3x” when asked

Conversion Complete

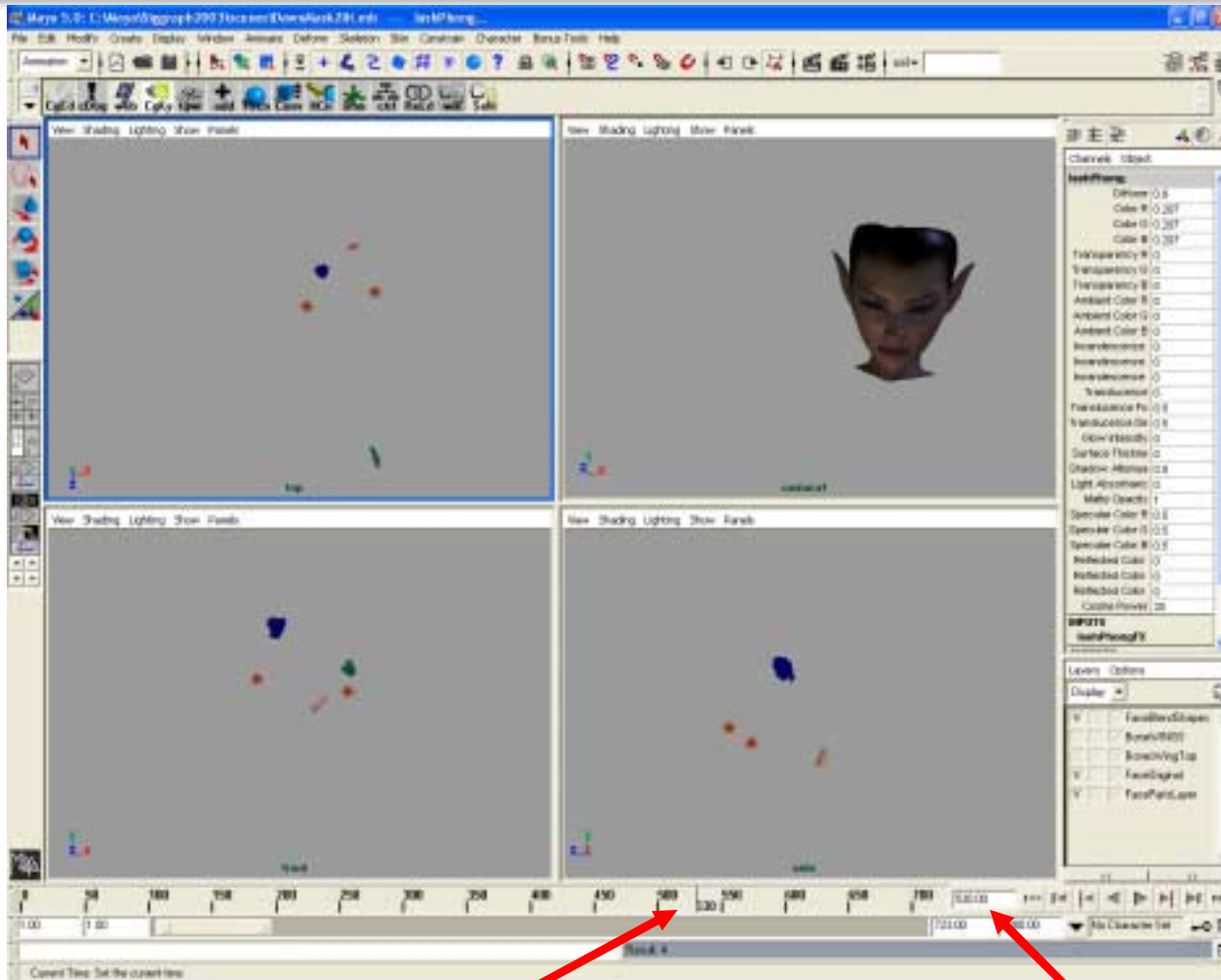


She Looks Darker



- Because now she matches the Maya software renderer

Different Frames (Frame 530)

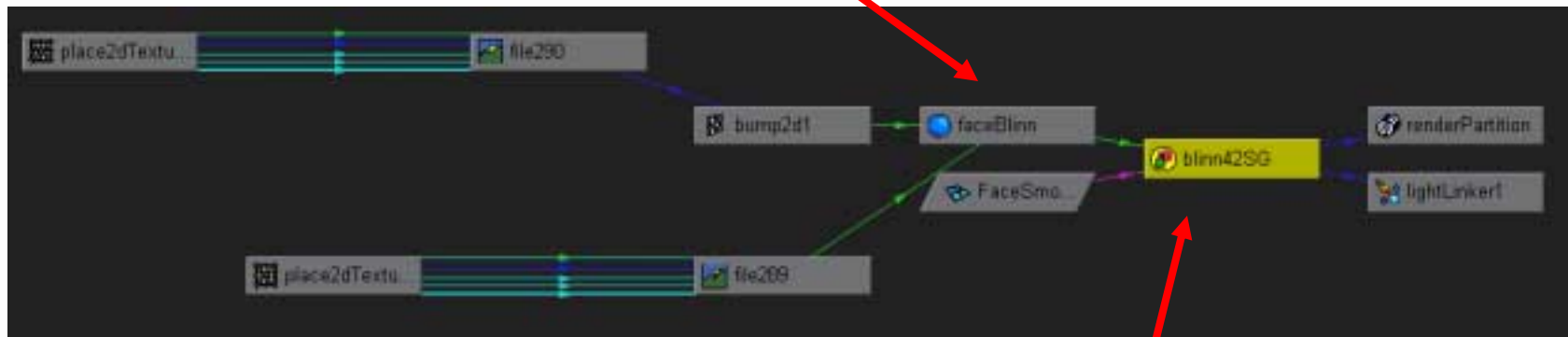


Playback
Slider

Frame
Number

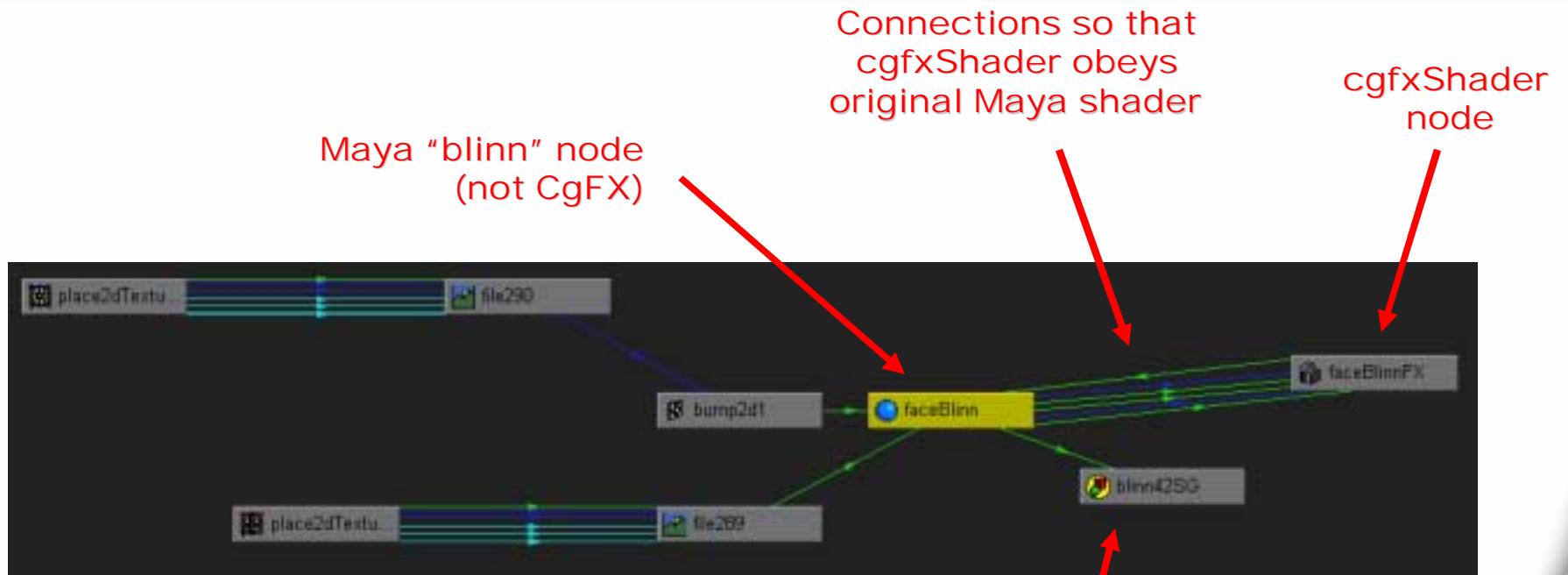
Linkages for Converted Shading Groups

Maya "blinn" node
(not CgFX)

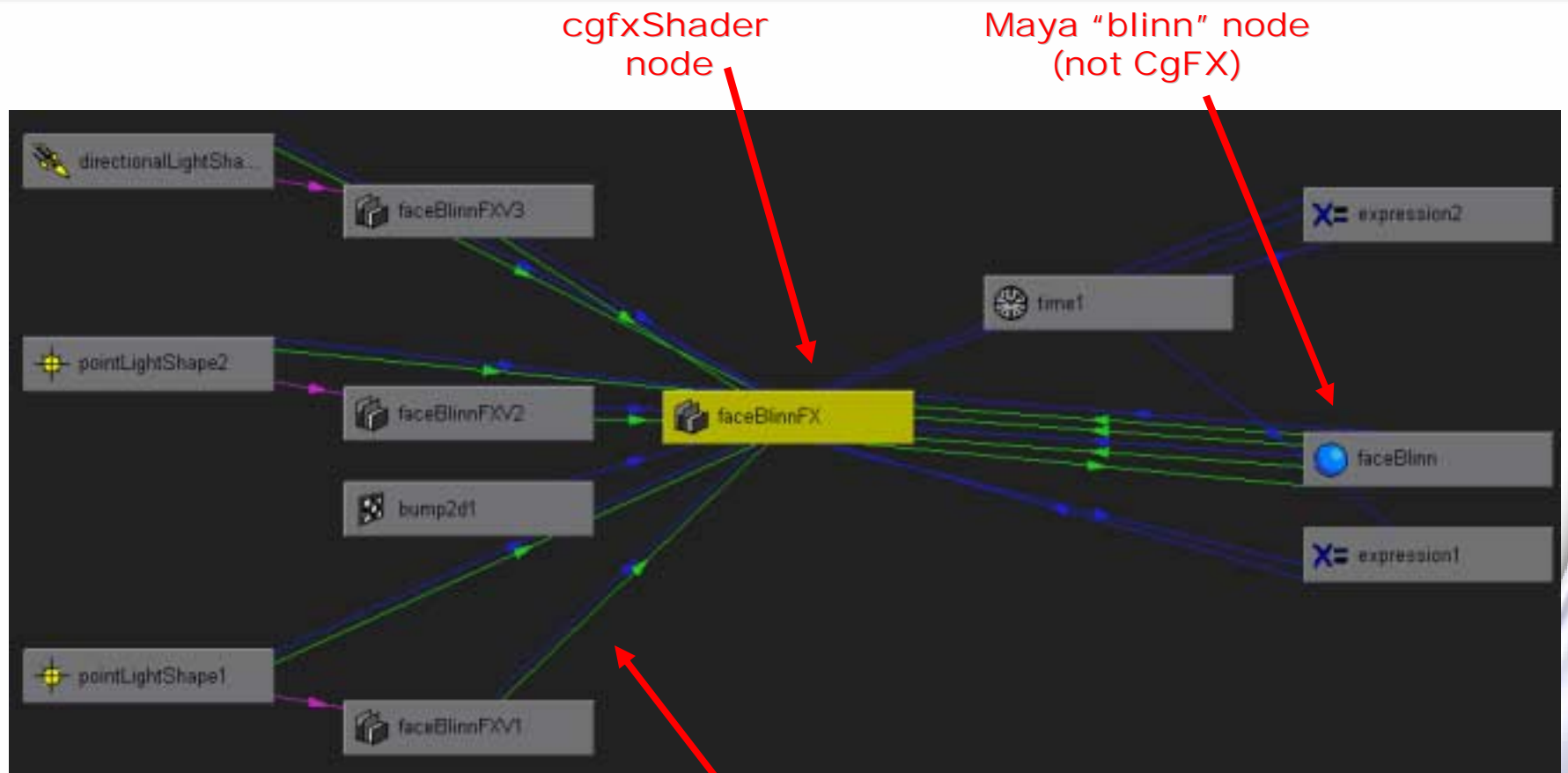


Shading
Group

Linkages for Maya blinn node "faceBlinn"



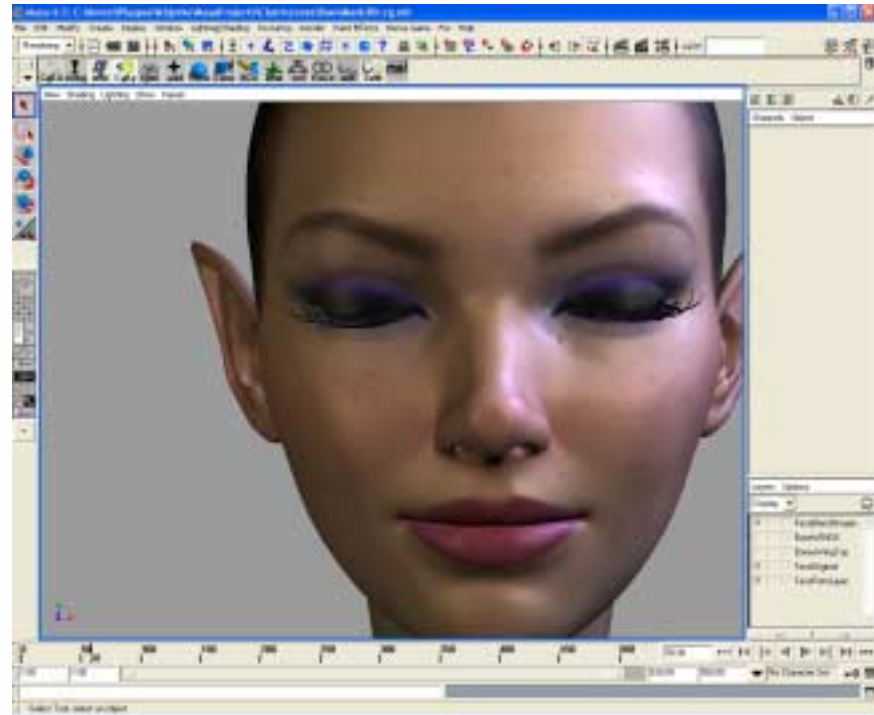
Auto-Created Linkages for "faceBlinnFX"



lights, directions, texture maps, etc - all with tracking connections to follow Maya shader

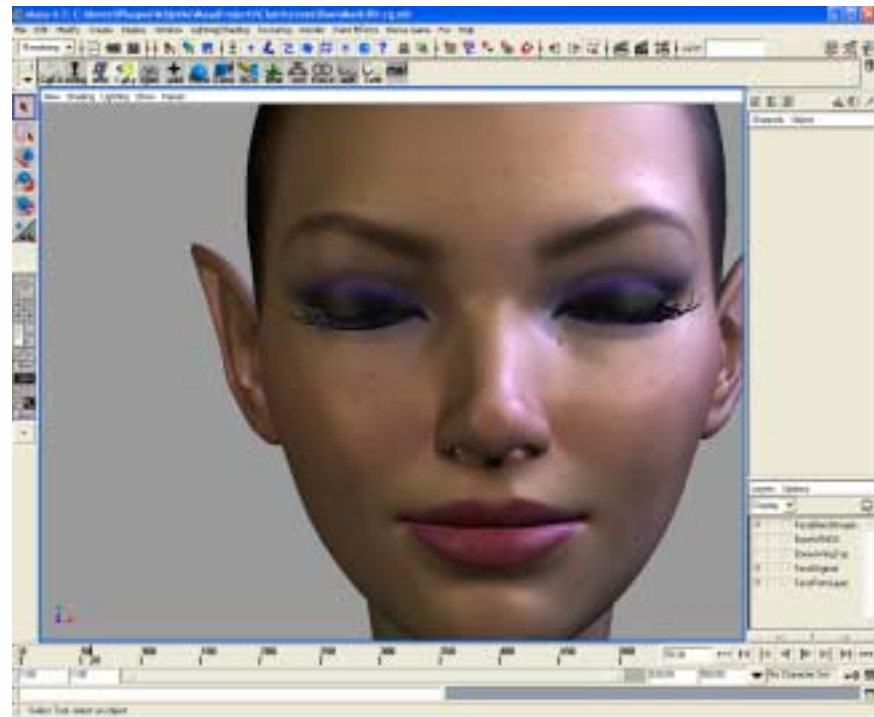
Playtime!

- Questions?
- Tweaking the scene:
 - Try moving lights around
 - Try changing light colors and intensities
 - Open other scenes
 - Hit “ReLd” to see nv2x versions
 - Look at generated shaders



Displacement Mapping, Vertex Skinning, and Animation

- Indexed Skinning
- Blendshapes
- Blobbies
- FFDs



Render to Texture and Special Multi-Pass Effects

- Render to Texture
 - Motion Blur
 - Depth of Field
- Stencil Shadows **[Better to call them Shadow Volumes I think]**
- Projective Texture, Including Shadow Maps



Exercise 2: Image Processing

- **Work with a graphic in real time by extending Exercise 1 or trying a new tool.**
 - **Take a CgFX file and adjust the colors of the graphic.**
 - **Try out OpenEXR**
- **Key:**
 - **How Cg is useful in many ways**
 - **Offers color correction**
 - **Shows how quickly you can change things**
 - **Shades existing graphic in real time**
- **Compare it to experiences you have had with other environments.**