

Using Real time Shaders

Tutorial - Summer 2005 – Jean-Marc Gauthier

Shaders create astonishing visual effects, for example transparencies would tax the computer's hardware resources with traditional rendering techniques. Shaders are also fast and allow user control over many parameters on the fly. This tutorial shows how to use real time shaders to render a 3D model of the heart. More information about coding and editing shaders can be found in Virtools help files > Getting Started, Shaders - Sample - Basic .

We explore the 3D visualization of the heart using shaders creating effects that would not be available with traditional animation techniques and rendering.

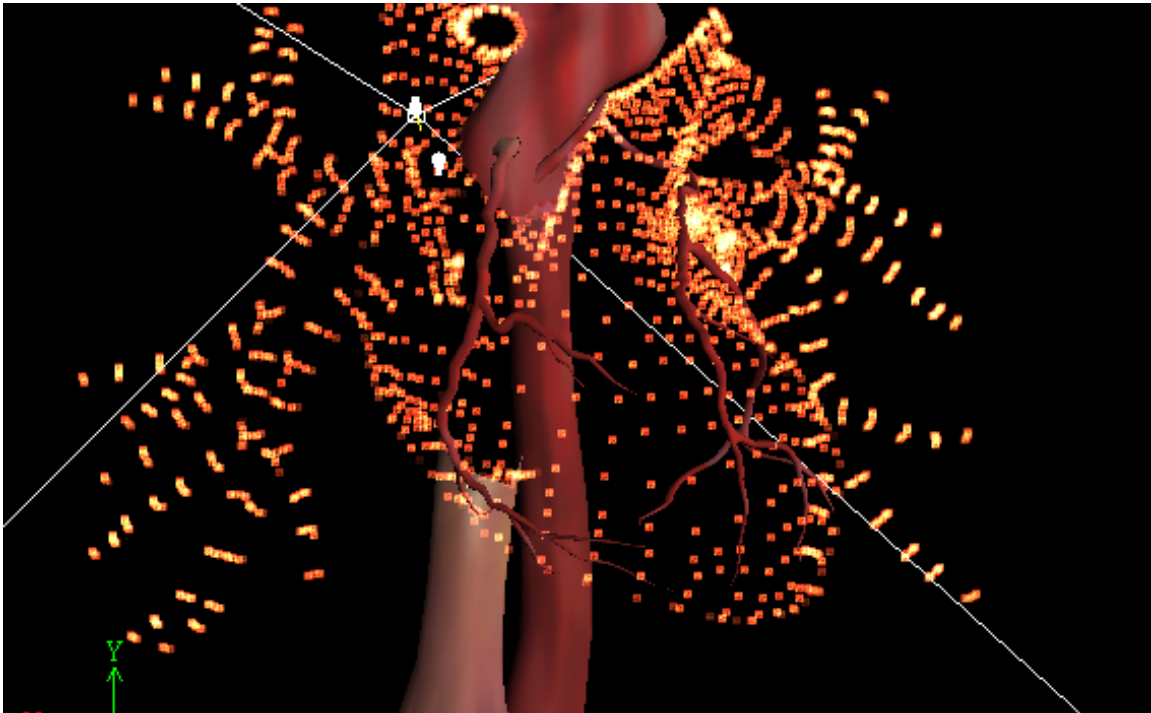
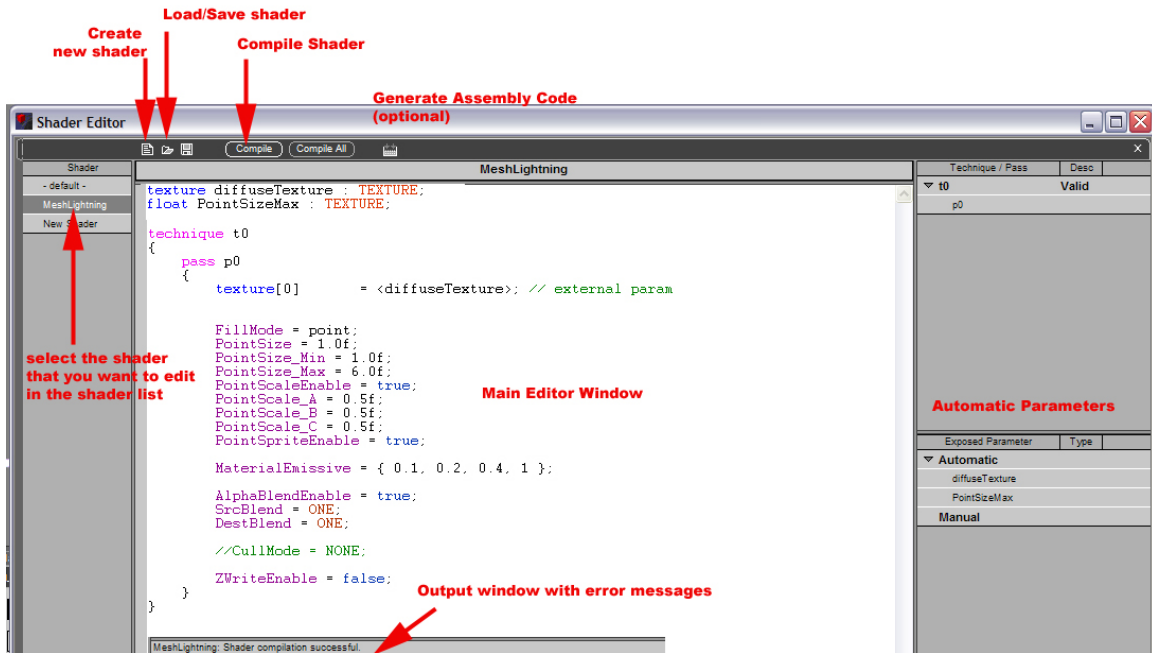


Illustration of the MeshLighting shader applied to the heart

What is a shader?

- A piece of code saved with the .fx extension, based exclusively on the DirectX 9 API and its high-level shader language (HLSL). DirectX 9 (or higher) must be installed on your computer; otherwise, you will not be able to compile Shaders in the Virtools Shader Editor or see any shader rendering.
- DirectX 9 is selected as the current Virtools rasterizer/rendering device
go to *Options > General Preferences > 3D Layout - Rendering > Rendering Device*.
- A Shader Compliant Graphics Card (GPU capable of supporting HLSL) and its most recent drivers are installed
Your 3D graphics card must be Shader compliant, i.e. capable of supporting vertex and pixel shaders. Make sure that your video card supports pixel shader version (2.0, 1.1, etc.)



For example, top the MeshLighting shader viewed inside the Virtool's Shader Editor. Bottom, the code for the MeshLighting shader

```

texture diffuseTexture : TEXTURE;
float PointSizeMax : TEXTURE;

technique t0
{
    pass p0
    {
        texture[0] = <diffuseTexture>; // external param

        FillMode = point;
        PointSize = 1.0f;
        PointSize_Min = 1.0f;
        PointSize_Max = 6.0f;
        PointScaleEnable = true;
        PointScale_A = 0.5f;
        PointScale_B = 0.5f;
        PointScale_C = 0.5f;
        PointSpriteEnable = true;

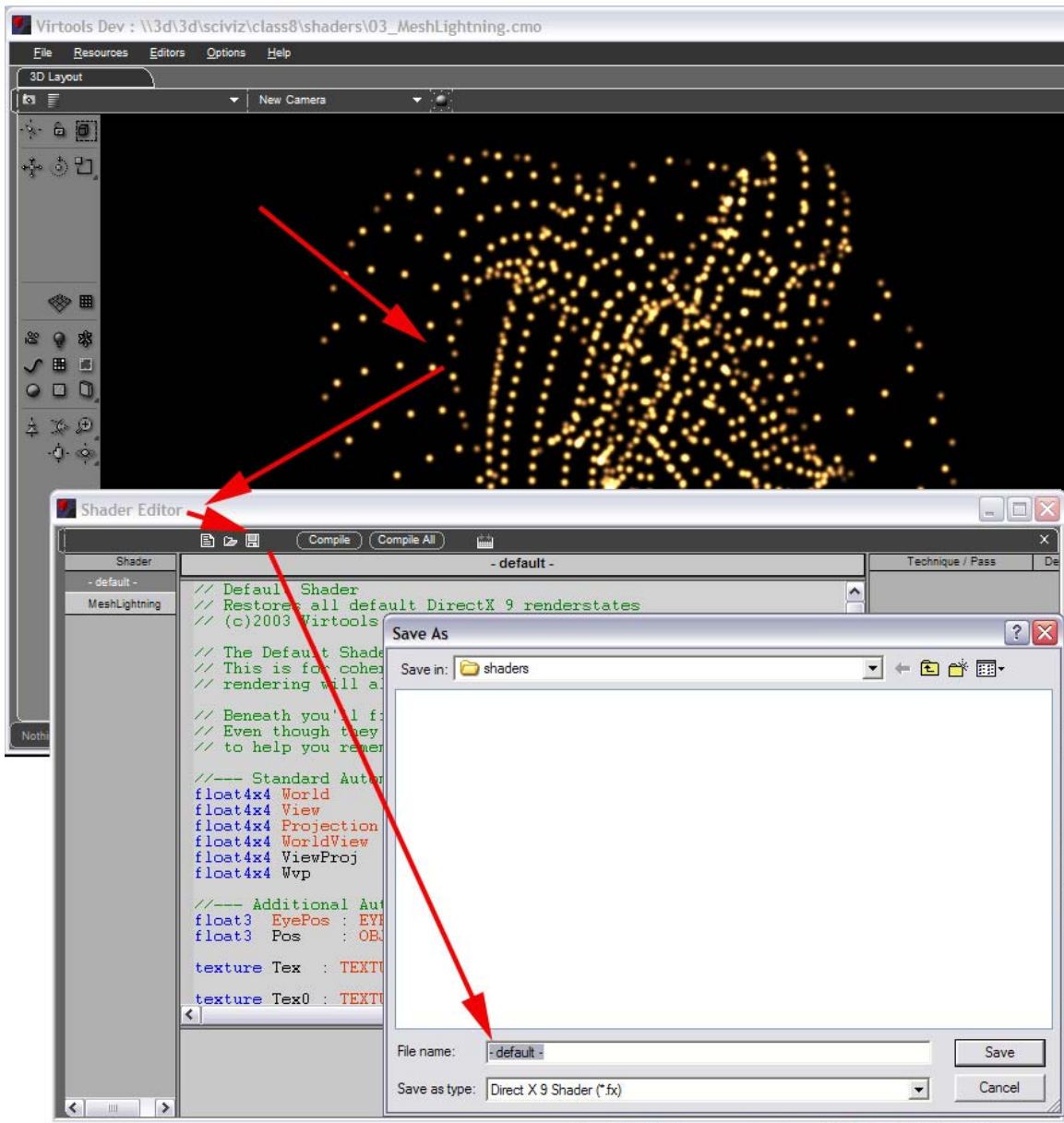
        MaterialEmissive = { 0.1, 0.2, 0.4, 1 };

        AlphaBlendEnable = true;
        SrcBlend = ONE;
        DestBlend = ONE;

        //CullMode = NONE;

        ZWriteEnable = false;
    }
}

```

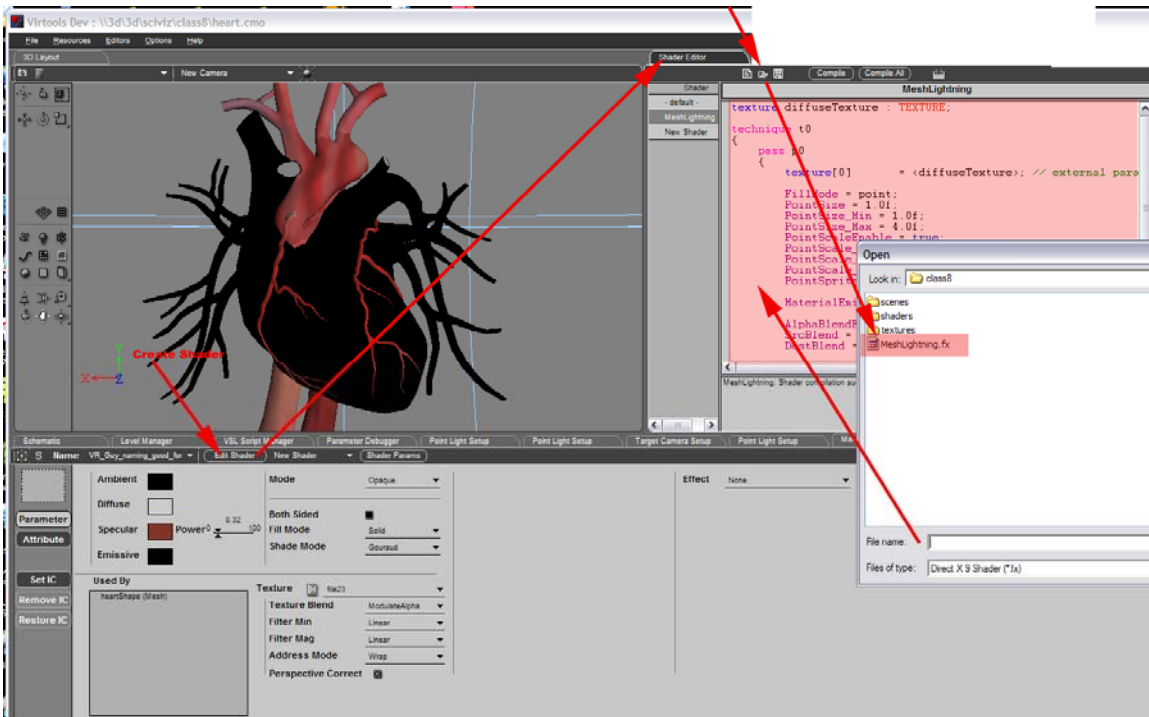


Top, Load the example of shader of your choice, for example MeshLightning.cmo, go to the Shader Editor and save the shader file as MeshLightning.fx. You can try with shader files that can be found online.

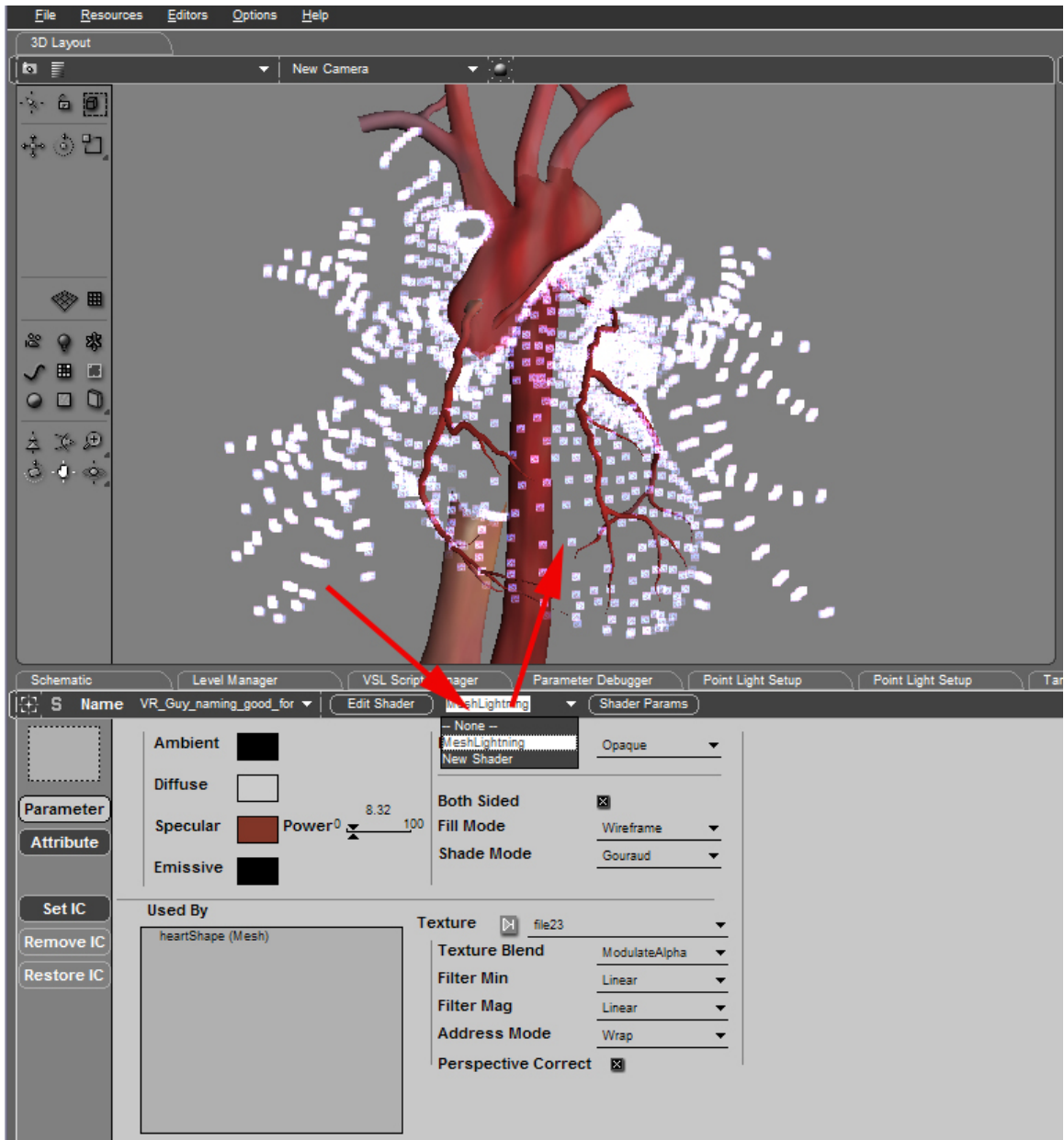
Bottom, let's apply the MeshLightning shader to the 3D model of the Heart.



Top, the heart exported from Maya as a 3D object. Please note that a heart with an animation will be exported as a Character. Bottom, select the heart's Material Setup > Create Shader. Go to the Shader Editor > open the MeshLighting.fx file.



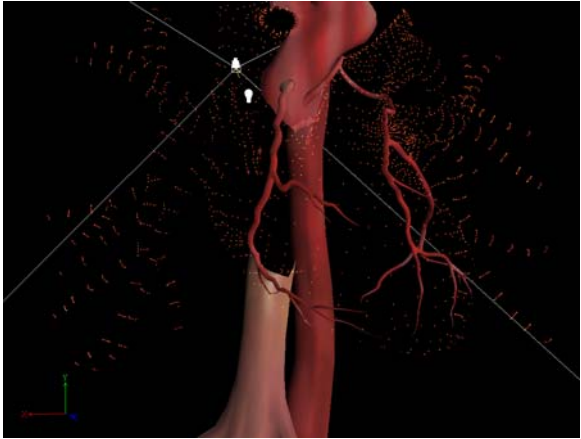
The text file is now ready for editing in the Shader Editor. Please note that you can load several shaders at the same time in the Shader Editor. The shaders will be listed by name on the left column. Editing and writing simple code for shader is a great programming exercise. Please note that there is an automatic keyword completion when you hit **Ctrl+Enter**. This works for types, semantic, states, enums, etc.



Hit Compile before leaving the Shader Editor

Go to the Heart's Material editor, select MeshLighting in the pull down menu on the right of "Edit Shader". Click on Edit Shader, the heart with the new shader shows up in the 3D Layout window.

Please note the changes of the heart according to new values of the shader's parameters.



Top, edit `PointSize_Max = 10.0f`; in order to change the size of the vertices. Bottom edit `MaterialEmissive = { 0.3, 0.2, 0.1, 1 }` to change the color

