



OpenGL Shader Designer Manual 1.5

12 June 2004

Introduction

The future of graphics programming is here and we got the tools you need. We present the **OpenGL Shader Designer**. A shader development IDE created for writing vertex and fragment (pixel) shaders in the OpenGL Shading Language (GLSL). This is the official manual for the IDE. We have written this as a guide that takes you through the steps of writing a simple multi-texture shader. It will explain everything from adding textures to passing uniforms and using varying variables. The goal of this manual is not to learn how to write shaders, it's to learn using the IDE. To learn more about the actual shaders, check the tutorials. They are available from the OpenGL Shader Designer group on the start menu and the Help - Tutorials menu in the IDE.

We have included lots of resources to aid you in your shader development. Textures are located in the "textures folder", material presets in "materials", light presets in "lights", glsl docs in "docs" and some sample shaders from 3d Labs in the "shaders" folder.

When installing the OpenGL Shader Designer, a new program group is created on your start menu. This group contains shortcuts to the program executable, this manual and the tutorials.

This manual will be updated on a regular basis, so check the site every now and then. There screenshots are not guaranteed to be from the latest build, but the differences should not be big.

We've included a feature to get info from the current driver. Use this to inspect your specs (if you submit bugs, this info would be great). You'll find it under Help -> Driver caps.

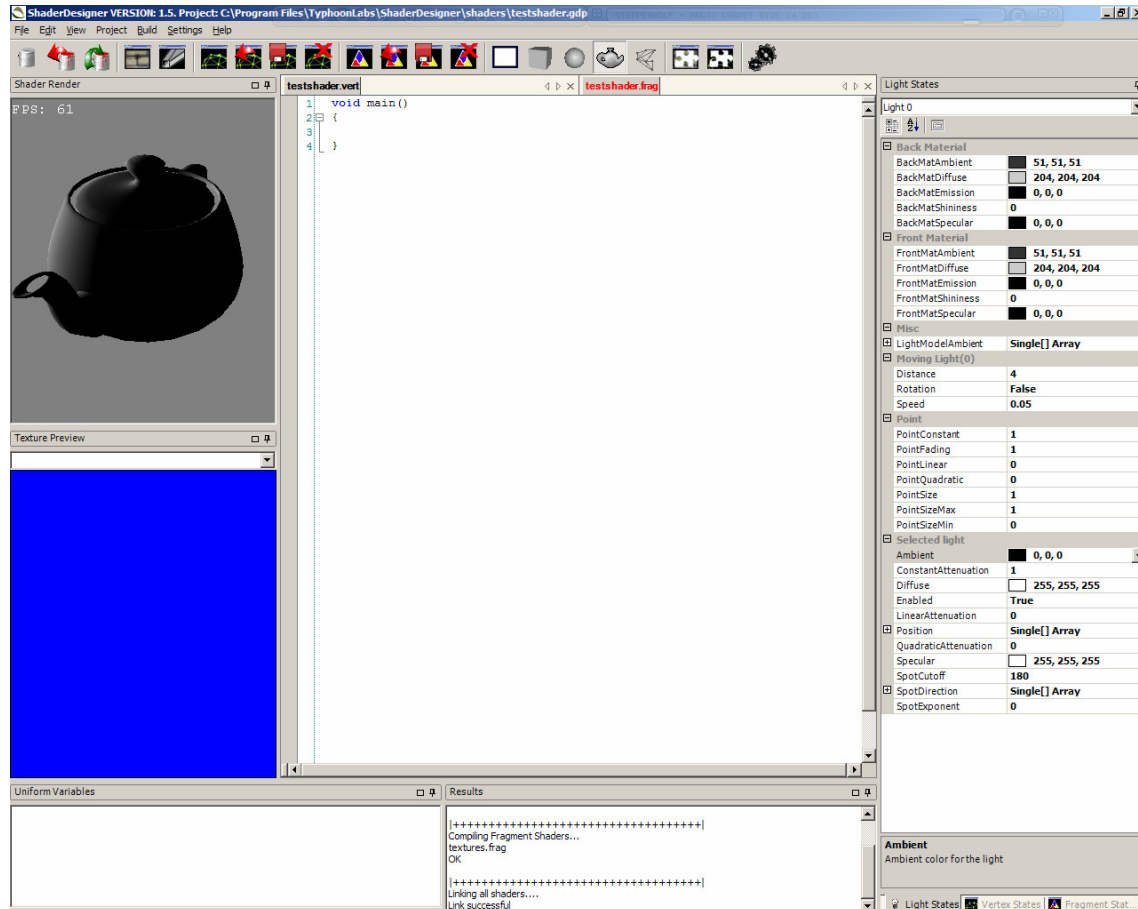
If you have any comments or suggestions, please send them to bugreport@typhoonlabs.com.

12 June 2004

Creating a new project and getting to know the IDE

To create a new project, select New from the File menu. Type in the name you want for your project and save it. The Shader Designer creates a fragment shader and a vertex shader for you.

You should now be looking at something like this:



The IDE with a new project loaded.

This is the main IDE. Every part of the editor is placed in it's own panel. You can attach and detach each single one and place them where you see fit.

You've got access to OpenGL states, material and light settings to your right. They are hid in sliding panels, so just move your cursor over them to see them. To change a value in the grid, you'll just type in a new value in the appropriate field. A special note about colors and alpha values: When setting a alpha value on a color in the grid you'll have to do it manually at the moments. Just add the alpha component before the three others. IE: you have a red color {255, 0, 0} and want to set the alpha to 50%. Change it to {128, 255, 0, 0}. We know that this isn't the best way to do this, and we are working on a color editor that makes this much easier.

In the middle of the screen you'll find the code box. This is where you'll write the actual shader code. At the top of the code window you'll see some tabs. This is an overview over the shader files in the current project.

In the upper left corner you'll see the preview pane. It gives a real-time preview of the shader you are working on. You can change the view by using the mouse. Press down the left mouse button and drag to rotate, press the right and drag to zoom. If you want smooth movement, hold down shift while moving. Lights are represented by small spheres colored with the diffuse component assigned to that light source.

Right under the preview pane you'll see texture previews. It's just a simple overview over the textures assigned to the current shader project. A combo box let's you choose texture unit. To bring up texture properties for the selected texture unit, just double-click the preview.

The uniform variable overview is located under the texture preview pane. Uniforms are variables sent from the host application (you'll find more info on those in the tutorials). Right clicking this window brings up a context menu with a few choices. You can add uniforms by clicking the "Add Uniform" button. Select type, name and value(s), and the optional widget. Once you press OK it will be added to the list. To modify an existing variable, highlight the one you want to edit and select "Modify Uniform". To delete it, press "Delete Uniform".

We've added widgets for uniforms in version 1.5. To utilize them, just select type in the create / modify dialog. We have two different kinds of widgets in this release. First out is the Color Slider. It let's you change a color uniform and get real-time previews. Double-click the color preview to bring up the standard color picker. Slide bar is a customizable slider for scalar values. You can set minimum value, maximum value and step. Changes to the passed uniform are updated real-time just like the color slider.

Under the uniform view you have the compilation output window. It will tell you about errors during shader compilation or linking. To see the line that causes the error, just double-click on the error line in the log, and the cursor will be moved to the line.

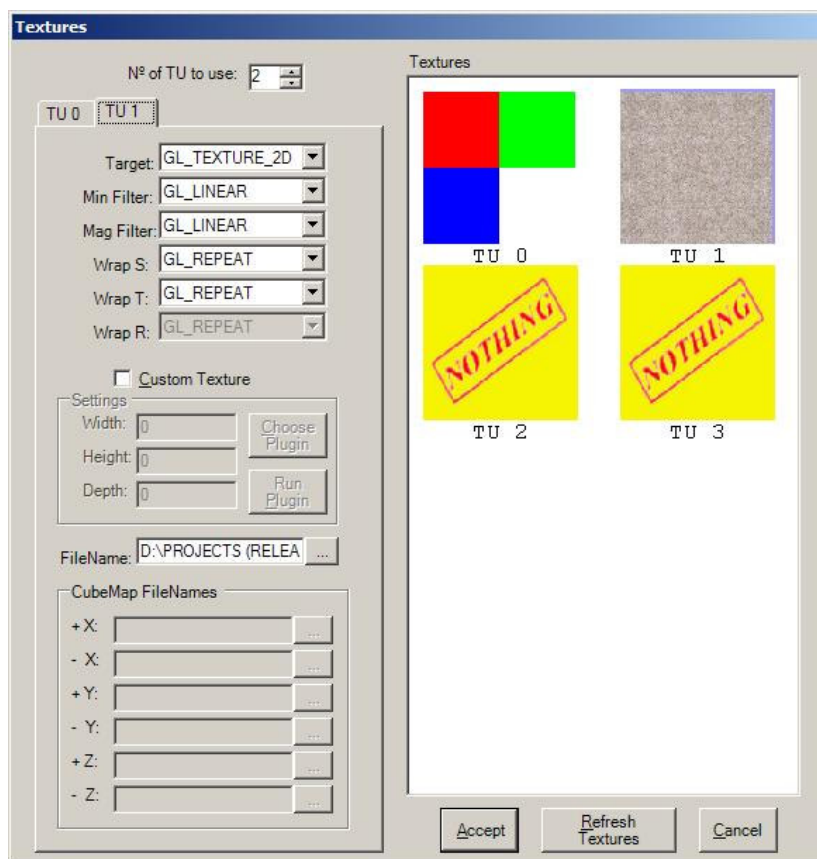
The View menu has settings that apply for shader preview. Here you can choose between different preview meshes, enable vertex normals in preview, preview the shader full-screen and take a snapshot. The snapshot will be stored in the folder called "snapshots".

The Project menu lets you manage texture units, perspective settings and add environment. The texture unit dialog is quite forward. Set the number of texture units you want to use, choose the files and press accept. The

Shader Designer automatically creates uniform samplers for the textures. The Perspective menu is also quite simple. Type in the field of view you want to use, set a aspect ratio and set the near and far clipping planes. The reset button sets the values to default. The Environment menu lets you add a background image or a skybox to the preview pane. This is useful when developing shaders that are partially transparent and so on. You can also disable shader rendering.

The build menu is not much to talk about. "Compile project" compiles and links the shaders in the project. "Clear compilation window" clears the output window.

The Help menu shows you this file, the about dialog, the tutorials and lets you decide how much tips you want on your tool tips.



Loading and using textures

Under the project menu you'll see "Textures...". This dialog is used to set up textures for your shader project. At the top you choose how many texture units you want to use. To assign a texture to the currently selected unit, press the [...] button that's located right after "FileName", or doubleclick on the preview box for the unit you are assigning a texture to. Once you've

located the file, accept and press "Refresh Textures" and check that the

correct texture got assigned to the correct texture unit. To scroll down press the arrow buttons. When you are all set, press Accept to return to the main window. Remember to add these lines to the vertex and/or pixel shader to gain access to the texture units.

```
uniform sampler2D TextureUnit0;  
uniform sampler2D TextureUnit1;
```

If you load any texture plug-ins they'll be available through the "Choose plugin" and "Run plugin" buttons. The output will be previewed just like textures you load.

Hotkeys

There are also hotkeys available for easy access to common dialogs and functions:

Ctrl + N	New project
Ctrl + O	Open project
Ctrl + S	Save project
Ctrl + Z	Undo
Ctrl + Y	Redo
Ctrl + X	Cut
Ctrl + C	Copy
Ctrl + V	Paste
Ctrl + T	Textures dialog
Ctrl + P	Perspective settings
Ctrl + E	Environment settings
Ctrl + M	Texture plug-ins
Ctrl + L	Attrib plugins
Ctrl + space	Intellisense
F2	Take snapshot
F4	Compile / Link

Writing and compiling a shader

Now we'll go through writing a new shader, adding textures and compiling it. Create a new project called "firstshader". Go add two textures to the project. Add "four.jpg" to texture unit one, and "circle.jpg" to unit two. Let the states be default.

Now you are ready to start writing your first shader. Select the shader file called "*firstshader.vert*" from the tabs above the code box. This is the vertex shader. The first thing we need to do is to create the main body of the shader. Type in the following:

```
void main(void)
{
}
```

Now add the following lines between the brackets.

```
gl_TexCoord[0] = gl_MultiTexCoord0;
gl_TexCoord[1] = gl_MultiTexCoord0;

gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
```

You've just set the texture coordinates for both texture units and multiplied the current vertex with the model-view projection matrix. That's all for the vertex shader.

Let's start at the pixel shader. Select if from the tabs and type in the following:

```
uniform sampler2D TextureUnit0;
uniform sampler2D TextureUnit1;

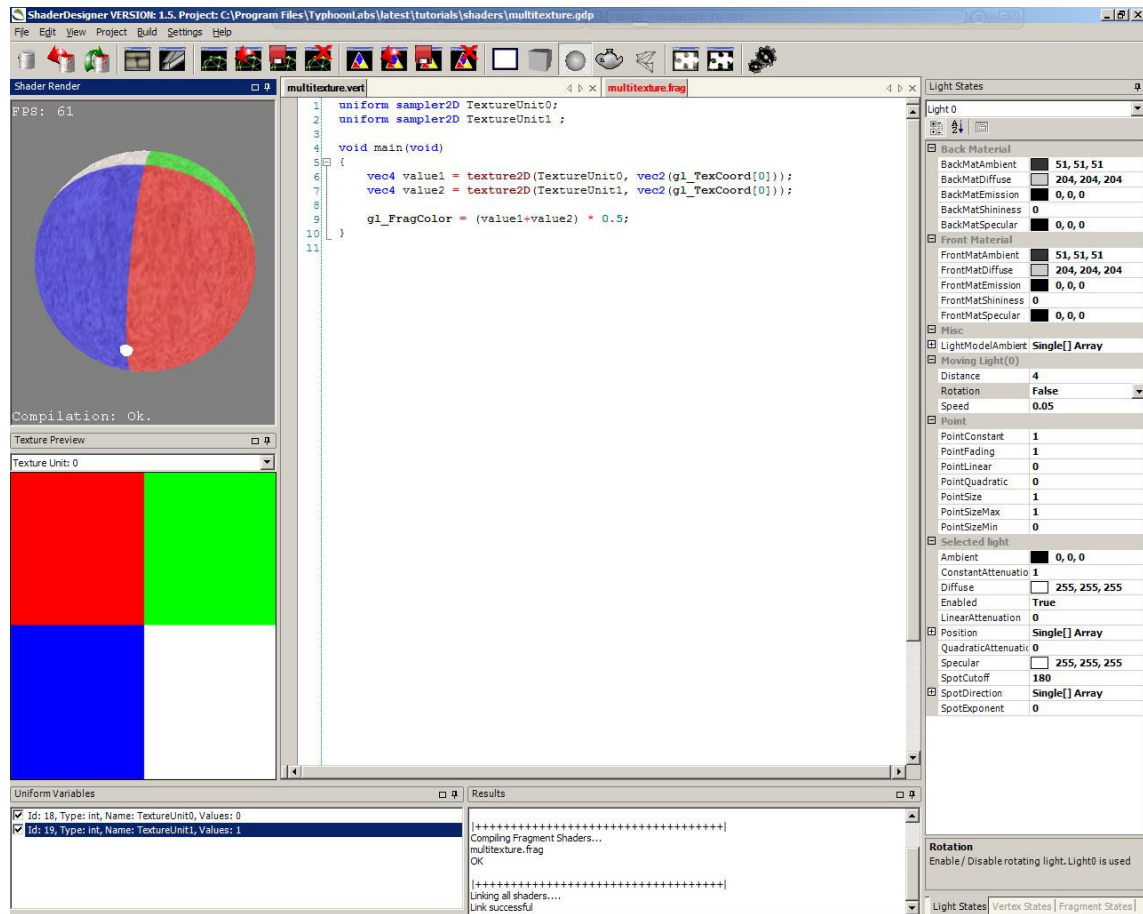
void main(void)
{
    vec4 value1 = texture2D(TextureUnit0, vec2(gl_TexCoord[0]));
    vec4 value2 = texture2D(TextureUnit1, vec2(gl_TexCoord[1]));

    gl_FragColor = (value1+value2) * 0.5;
}
```

That's all. The first two lines are samplers for the texture units. The vec4 lines extract the color of the current texture fragments, and the gl_FragColor line sets the color of the fragment being written. For a better explanation of the shader code, see the tutorials.

NOTE: The meshes we supply with the Shader Designer has only one set of texture coordinates. That's the reason why we use gl_TexCoord[0] when looking up in unit 1.
`vec4 value2 = texture2D(TextureUnit1, vec2(gl_TexCoord[0]));`

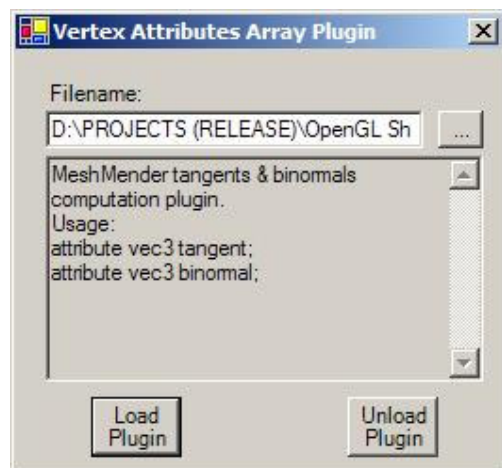
Press F4 to compile the project and update the preview pane. You should now see something like this:



The IDE with a multi-texture project loaded.

Plug-ins

The OpenGL Shader Designer has plug-in support for texture plug-ins and vertex attribute plug-ins. They are both accessed through the Plug-in menu under Project. To add a plug-in, simply locate the file and press "Add plug-in". When it comes to the usage of plug-ins it can be very different from plug-in to plug-in. Some are transparent; others provide dialogs and advanced GUI's. Check the documentation for the plug-in of current interest for information on how it works.



Using the included tangent / binormal plugin

Vertex Attrib plug-ins are stored as .dap files in the "Plugins" folder. To add a plug-in to the current project, open the Vertex Attrib Plugin dialog (CTRL + L). Press the [...] button and locate the file called *meshmender.dap*. Click "Open" and then "Load plugin". If everything succeeded you'll now have a short

description of the plug-in and its usage in the output. It's the plug-in authors responsibility to provide a understandable and informative description here. In this case, to access tangents and binormals you'll have to add "attribute vec3 tangent" and "attribute vec3 binormal" to your vertex shader code. These variables will automatically be populated by the plug-in.