

ShaderMaker

ShaderMaker is a source code template that can be used to compile [FreeframeGL](#) plugins from shader source copied from "[GLSL Sandbox](#)" and "[ShaderToy](#)".

Plugins can be created from these shaders by copy/paste of the fragment shader code into the *ShaderMaker.cpp* source file and re-compiling.

The code is pasted into a section of the source file which uses the [Stringizing](#) operator (`#`) which converts the shader code into a string which is then used by the shader compiler on load of the plugin.

There are some limitations of using the stringizing operator in this way because it depends on the `"#"` symbol, e.g. `#(.. code)`, therefore there cannot be any `#` characters in the code itself.

It is useful to create, select and test shaders with the *ShaderLoader* plugin first. Then they will compile successfully in *ShaderMaker*. Refer to the *ShaderLoader* documentation for details.

GLSL Sandbox and *ShaderToy* examples are included in the source.

Compilation

Windows

This is a Visual Studio C++ project that can make any number of them. Download everything and unzip into in any folder, open the VS2010 solution file with Visual Studio and change to "release", it should build OK as-is. More examples are in the source file.

Mac

In build/osx there are both an XCode project and a makefile. In both cases, the resulting bundle will be created in Binaries/osx.

How to use

To make your own shader plugin, all you do is copy/paste the shader code into the source file, change the plugin information and rename the resulting dll. There are some things to take note of, but hopefully the code and documentation are clear enough.

Compilation will result in a FreeframeGL plugin "ShaderMaker.dll. Rename the file as you require. You will already have entered the unique plugin information in the source code before compilation. If you don't give the plugin a unique ID, it will conflict with any other copies you may have made even if it is a different name.

The dll can then be copied into the plugin folder for the host application.

For "Resolume Avenue" or "Resolume Arena", you can copy them to the default plugin "vfx" folder. Or you can define specific folders for effects.

For "*Isadora*" copy to the Program files "..\Common Files\Freeframe" folder. Create this folder if it does not exist.

Magic allows "additional module folders" to be defined so using the "..\Common Files\Freeframe" folder is a good idea.

Operation

Although *ShaderMaker* results in an "effect" plugin, it will load without any texture input even if one is required by the shader, but it simply will not show anything. If a texture is not required by the shader, any input texture is ignored.

The plugin allows for two texture inputs. Hosts tested : [*Magic*](#), [*Isadora*](#), [*Resolume*](#)

For *Magic*, connect one or two texture inputs or none. The textures are used if the shader requires them or are ignored if not.

Resolume will also work with no texture source or with one or two sources and similarly the textures are either used or not depending on whether the shader requires them.

For *Isadora*, the plugin will not work without a texture input. Both texture inputs have to be connected to a texture source.

Changing the speed

The "Speed" control allows the animation rate of the shader to be controlled in the same way as for the speed controls in *GLSL Sandbox* or *ShaderToy*. *ShaderMaker* time is calculated internally rather than using FFGL time input from the host.

Position control

ShaderMaker does not react on mouse movement, but parameters can be used in an equivalent way to the Mouse input variables within the shader.

Mouse X & Mouse Y are passed through from the first two parameters of the plugin.

A further two "mouse" coordinates are passed through from the next two parameters of the plugin Mouse Left X and Mouse Left Y. These are equivalent to the left button pressed.

Colour control

An additional uniform for colour input can be used. The values do not need to be used for colour, so the four floats are available for other purposes.

```
uniform vec4 inputColour; // (red, green, blue, alpha)
```

This uniform is added internally for a ShaderToy shader file, but for a GLSL Sandbox file it has to be entered into the file itself.

Contact, Credits and License

Contact

Contact us at spout.zeal.co

Credits

Implementation by Lynn Jarvis leadedge@adam.com.au. MacOSX port by Amaury Hazan amaury@billaboop.com.

The MacOSX port was supported by Coldcut/Ninja Tune as a contribution to the Visuals Community

License

Copyright (C) 2015. Lynn Jarvis, Leading Edge. Pty. Ltd. (leadedge@adam.com.au)
Ported to OSX by Amaury Hazan (amaury@billaboop.com)

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You will receive a copy of the GNU Lesser General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.