Christie Digital Systems Germany GmbH
Richard-Byrd-Strasse 19
50829 Cologne
Germany

+ 49 221 99 512-0

pandorasbox@christiedigital.com

christiedigital.com

To whom it may concern,

August 2021

Pandoras Box V8 – Introduction to custom shader FXs

This how-to guide is targeted to developers with experience in HLSL shader programming and does not aim to explain how to write shaders in general. It shall provide a brief overview of specific details which are needed for *Pandoras Box* shaders.

The following pages will highlight some topics to better understand the provided shader examples.

As the number of video layers isn't limited anymore within V8, we are happy to also explain how to load external textures to create much more interesting shaders or simply combine multiple videos on a single layer as an overlay effect.

Contents

General information 3
Editing and loading custom FXs 4
Basic shader structure 6
User interface description 7
Shader code 9
    Example code “Media Overlay Add” 10
    Specific *Pandoras Box* variables 11

General information

Custom shaders for *Pandoras Box* need to be in the following format:

Shader language:	HLSL
Pixel shader:	5.0
File format:	XML
File extension:	.ccfx (custom Christie FX)

Pandoras Box loads shaders and textures from the following folders:

C:\Program Files\Christie\Pandoras Box 8.x.x\data\fx\shader

- custom shaders

C:\Program Files\Christie\Pandoras Box 8.x.x\data\fx\config

- special configurations like dropdown menus
- thumbnails for dropdown menus

C:\Program Files\Christie\Pandoras Box 8.x.x\data\fx\stockTextures

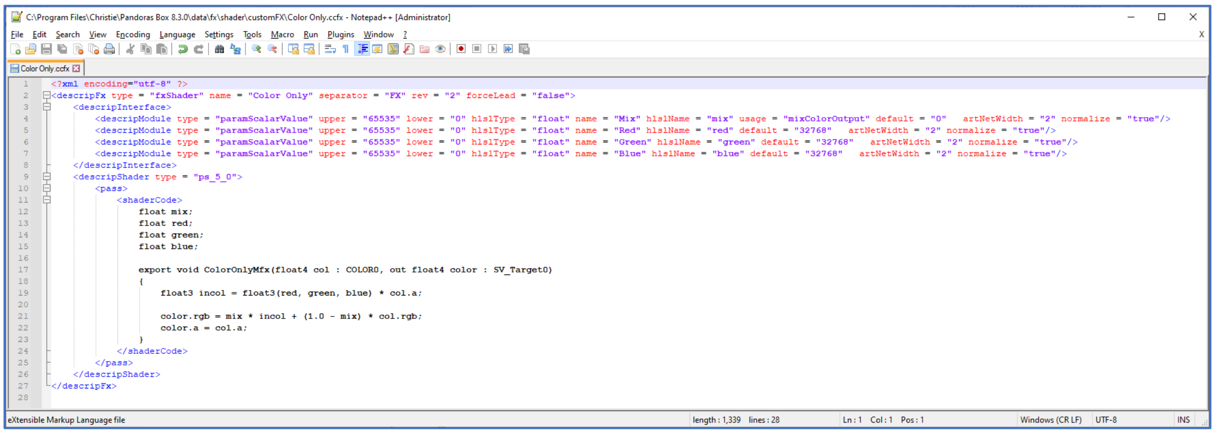
- fixed stock textures used inside a shader

Editing and loading custom FXs

We recommend using *Notepad++* as your editor for shaders due to multiple reasons.

Custom FXs are stored in the installation path of *Pandoras Box* which is a protected folder by default. You will therefore need administrator rights to change files inside it. *Notepad++* is able to be launched in Administrator mode so you can directly edit and save files in the protected environment.

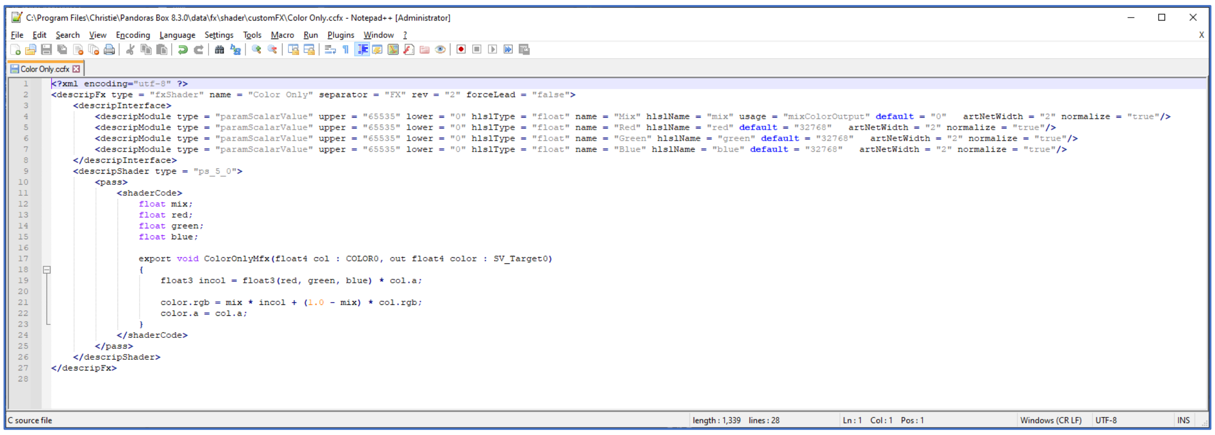
Furthermore, the application is very useful due to its syntax highlighting.
Depending on the area you are working in, we recommend to set *Notepad++* to either *XML*, when working on the user interface elements...


C:\Program Files\Christie\Pandoras Box 8.3.0\data\fx\shader\customFX\Color Only.ccfx - Notepad++ [Administrator]

```
<?xml encoding="utf-8" ?>
<descripFx type = "fxShader" name = "Color Only" separator = "FX" rev = "2" forceLead = "false">
    <descripInterface>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Mix" hlslName = "mix" usage = "mixColorOutput" default = "0"   artNetWidth = "2" normalize = "true"/>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Red" hlslName = "red" default = "32768"   artNetWidth = "2" normalize = "true"/>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Green" hlslName = "green" default = "32768"   artNetWidth = "2" normalize = "true"/>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Blue" hlslName = "blue" default = "32768"   artNetWidth = "2" normalize = "true"/>
    </descripInterface>
    <descripShader type = "ps_5_0">
        <pass>
            <shaderCode>
                float mix;
                float red;
                float green;
                float blue;

                export void ColorOnlyMfx(float4 col : COLOR0, out float4 color : SV_Target0)
                {
                    float3 incol = float3(red, green, blue) * col.a;

                    color.rgb = mix * incol + (1.0 - mix) * col.rgb;
                    color.a = col.a;
                }
            </shaderCode>
        </pass>
    </descripShader>
</descripFx>
```

eXtensible Markup Language file — length : 1,339 lines : 28 — Ln : 1 Col : 1 Pos : 1 — Windows (CR LF) — UTF-8 — INS

...or *C* when actually coding the shader.


C:\Program Files\Christie\Pandoras Box 8.3.0\data\fx\shader\customFX\Color Only.ccfx - Notepad++ [Administrator]

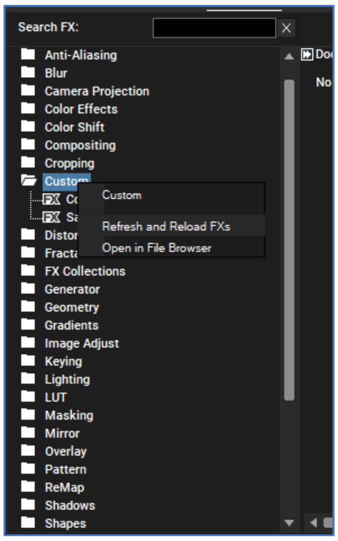
```
<?xml encoding="utf-8" ?>
<descripFx type = "fxShader" name = "Color Only" separator = "FX" rev = "2" forceLead = "false">
    <descripInterface>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Mix" hlslName = "mix" usage = "mixColorOutput" default = "0"   artNetWidth = "2" normalize = "true"/>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Red" hlslName = "red" default = "32768"   artNetWidth = "2" normalize = "true"/>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Green" hlslName = "green" default = "32768"   artNetWidth = "2" normalize = "true"/>
        <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Blue" hlslName = "blue" default = "32768"   artNetWidth = "2" normalize = "true"/>
    </descripInterface>
    <descripShader type = "ps_5_0">
        <pass>
            <shaderCode>
                float mix;
                float red;
                float green;
                float blue;

                export void ColorOnlyMfx(float4 col : COLOR0, out float4 color : SV_Target0)
                {
                    float3 incol = float3(red, green, blue) * col.a;

                    color.rgb = mix * incol + (1.0 - mix) * col.rgb;
                    color.a = col.a;
                }
            </shaderCode>
        </pass>
    </descripShader>
</descripFx>
```

C source file — length : 1,339 lines : 28 — Ln : 1 Col : 1 Pos : 1 — Windows (CR LF) — UTF-8 — INS

Choose *Language* from the menu and select the desired language accordingly.

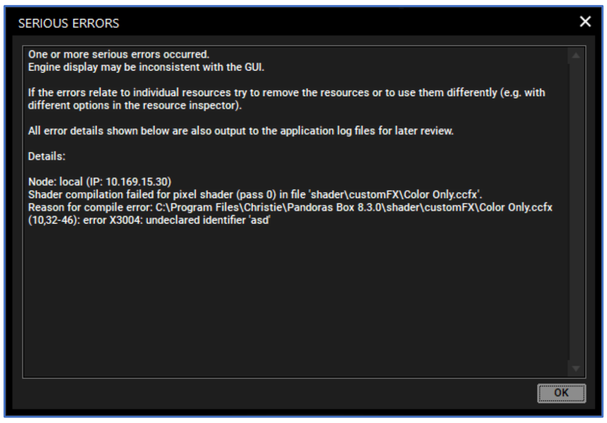
When working on shaders and trying out their visual results in *Pandoras Box*, you can reload shaders after editing those without the need of restarting the application.

Simply navigate to the according folder within the *Aeon FX* tab inside *Pandoras Box* and choose *Refresh and Reload FXs* from the folder's context-menu.



Please note that only the shader code itself can be reloaded. Changes to user interface elements require the FX to be deleted and reapplied to the layer.

In case of errors in your code, *Pandoras Box* will report the compile error with a popup and tell the line of the issue. The lines refer to the shader code only, not the absolute line number within the XML file.



```
SERIOUS ERRORS

One or more serious errors occurred.
Engine display may be inconsistent with the GUI.

If the errors relate to individual resources try to remove the resources or to use them differently (e.g. with
different options in the resource inspector).

All error details shown below are also output to the application log files for later review.

Details:

Node: local (IP: 10.169.15.30)
Shader compilation failed for pixel shader (pass 0) in file 'shader\customFX\Color Only.ccfx'.
Reason for compile error: C:\Program Files\Christie\Pandoras Box 8.3.0\shader\customFX\Color Only.ccfx
(10,32-46): error X3004: undeclared identifier 'asd'

OK
```

Basic shader structure

As mentioned above, the custom shader code is wrapped into an XML structure and the layout is depicted below:

```
<?xml encoding=”utf-8” ?>
<descripFx type = “fxShader” name = “FX Name” separator = “FX” rev = “2” forceLead = “false”>
    <descripInterface>
        // Define user interface elements here.
    </descripInterface>
    <descripShader type = “ps_5_0”>
        <pass>
            <vertexSet scaleW="0.9" scaleH="0.9"/>
            <blendMode>
                <color blendOp="ADD" srcBlend="ONE" destBlend="ZERO"/>
                <alpha blendOp="ADD" srcBlend="ONE" destBlend="ZERO"/>
            </blendMode>
            <shaderCode>
                //Shader code is placed here for first render pass.
            </shaderCode>
        </pass>
        <pass repeats=”5”>
            <shaderCode>
                //Shader code is placed here for following 5 render passes.
            </shaderCode>
        </pass>
    </descripShader>
</descripFx>
```

Replace *FX Name* for the desired shader name which shall be displayed in *Pandoras Box*.

rev=”2” needs to be set in order to use the depicted format of render passes

forceLead=”true” needs to be used when an interim texture shall be rendered. When lookups are required in an FX, all other FXs prior to it in the chain will then be rendered to allow the current FX to sample the combined result of all prior FXs. Default is *“false”* and shall only be used when needed.

<pass> encloses the shader code per render pass in the order of processing. The tag only needs to be set for multi-pass shaders.

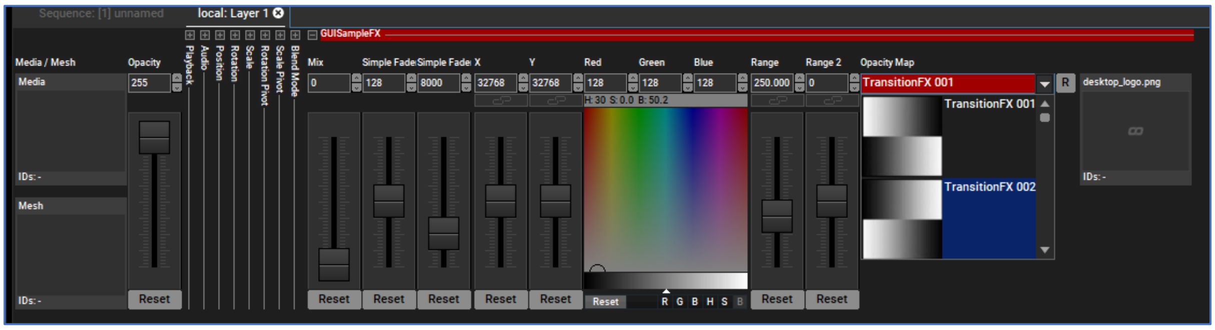
<vertexSet scaleW/H> scales the texture to the given percentage

<blendMode> allows for different blend modes to be used within the shader for color and alpha

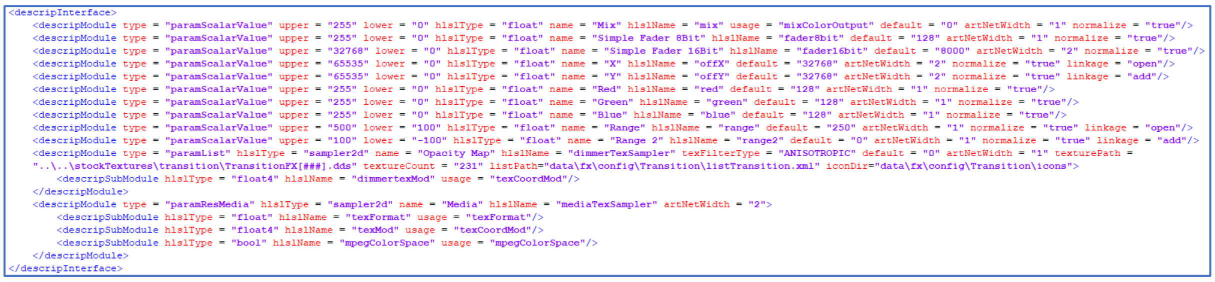
User interface description

The *<descripInterface>* defines the elements which are made available to the user or are used in the background. Max values are defined along with channel width, names, defaults and other values.

The below definition will result into the following user interface representation:



These definitions have been made:



```
<descripInterface>
    <descripModule type = "paramScalarValue" upper = "255" lower = "0" hlslType = "float" name = "Mix" hlslName = "mix" usage = "mixColorOutput" default = "0" artNetWidth = "1" normalize = "true"/>
    <descripModule type = "paramScalarValue" upper = "255" lower = "0" hlslType = "float" name = "Simple Fader 8Bit" hlslName = "fader8bit" default = "128" artNetWidth = "1" normalize = "true"/>
    <descripModule type = "paramScalarValue" upper = "32768" lower = "0" hlslType = "float" name = "Simple Fader 16Bit" hlslName = "fader16bit" default = "8000" artNetWidth = "2" normalize = "true"/>
    <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "X" hlslName = "offX" default = "32768" artNetWidth = "2" normalize = "true" linkage = "open"/>
    <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Y" hlslName = "offY" default = "32768" artNetWidth = "2" normalize = "true" linkage = "add"/>
    <descripModule type = "paramScalarValue" upper = "255" lower = "0" hlslType = "float" name = "Red" hlslName = "red" default = "128" artNetWidth = "1" normalize = "true"/>
    <descripModule type = "paramScalarValue" upper = "255" lower = "0" hlslType = "float" name = "Green" hlslName = "green" default = "128" artNetWidth = "1" normalize = "true"/>
    <descripModule type = "paramScalarValue" upper = "255" lower = "0" hlslType = "float" name = "Blue" hlslName = "blue" default = "128" artNetWidth = "1" normalize = "true"/>
    <descripModule type = "paramScalarValue" upper = "500" lower = "100" hlslType = "float" name = "Range" hlslName = "range" default = "250" artNetWidth = "1" normalize = "true" linkage = "open"/>
    <descripModule type = "paramScalarValue" upper = "100" lower = "-100" hlslType = "float" name = "Range 2" hlslName = "range2" default = "0" artNetWidth = "1" normalize = "true" linkage = "add"/>
    <descripModule type = "paramList" hlslType = "sampler2d" name = "Opacity Map" hlslName = "dimmerTexSampler" texFilterType = "ANISOTROPIC" default = "0" artNetWidth = "1" texturePath =
    "..\..\stockTextures\transition\TransitionFX[###].dds" textureCount = "231" listPath="data\fx\config\Transition\listTransition.xml" iconDir="data\fx\config\Transition\icons">
        <descripSubModule hlslType = "float4" hlslName = "dimmertexMod" usage = "texCoordMod"/>
    </descripModule>
    <descripModule type = "paramResMedia" hlslType = "sampler2d" name = "Media" hlslName = "mediaTexSampler" artNetWidth = "2">
        <descripSubModule hlslType = "float" hlslName = "texFormat" usage = "texFormat"/>
        <descripSubModule hlslType = "float4" hlslName = "texMod" usage = "texCoordMod"/>
        <descripSubModule hlslType = "bool" hlslName = "mpegColorSpace" usage = "mpegColorSpace"/>
    </descripModule>
</descripInterface>
```

The modules can be defined with the following descriptions:

• *type*	UI element type
○ *paramScalarValue*	fader
○ *paramList*	dropdownlist
○ *paramResMedia*	texture sampler
• *upper*	max value
• *lower*	min value
• *hlslType*	HLSL variable type
• *name*	name of the parameter within the UI
• *hlslName*	name of the shader code variable linked to the parameter (needs specific naming for textures and samplers - see chapter "variables")
• *default*	default value when initialized or reset
• artNetWidth	number of assigned DMX channels
• *normalize = "true"*	normalizes values to 0..1 range (e.g.: 0..255 becomes 0..1 with 128 = 0.5)
• *linkage="open"*	creates new link group
• *linkage="add"*	adds parameter to the last opened link group
• *avoidGuiLoad = "always"*	parameter will not be visible in UI (Useful to load textures that users shall not assign.)
• *usage*	links engine parameters to shader parameters

Best practice:

There should always be a "Mix" fader set up like in the example code in order to be able to visually deactivate the shader.

Three faders in a row called *Red*, *Green* and *Blue* will automatically be converted into a color picker.

Variables names must not start with *pb* as those values are reserved for internal purposes.

Shader code

Pixel shaders work on a per pixel basis. This means a pixel shader code runs for every pixel in the image. Assuming an HD output at 60 Hz means executing the code 1920*1080*60 = 124.416.000 times per second.

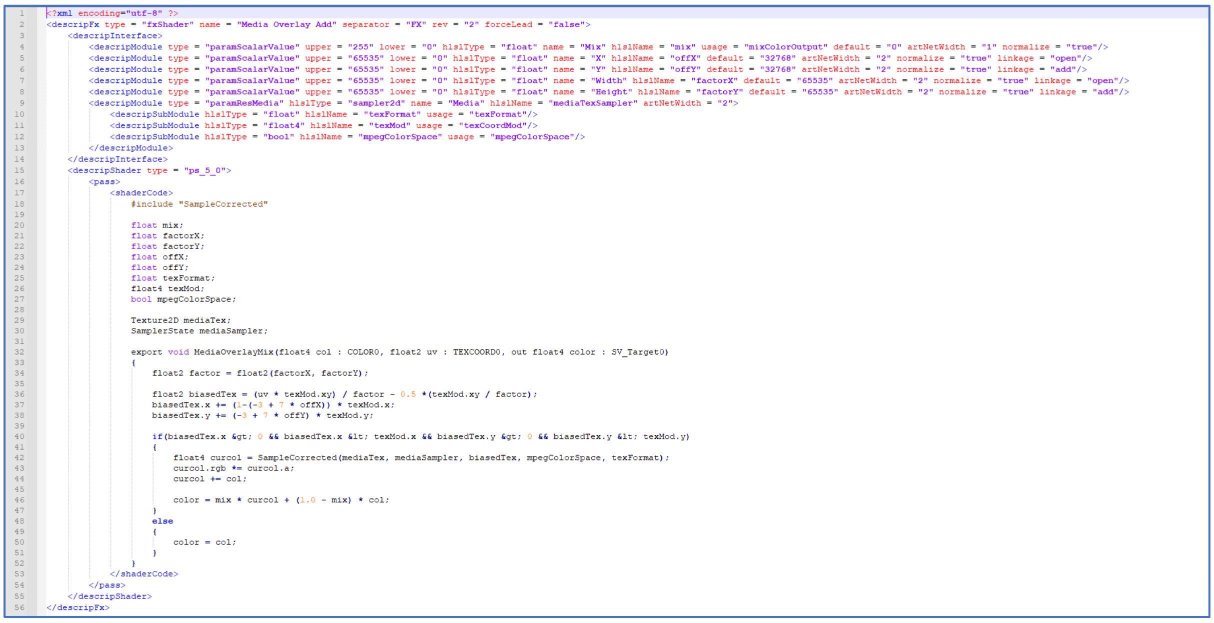
The shader code by itself has knowledge of:

- the position of the pixel being calculated
- the original pixel color being calculated
- the shader pass

Since the HLSL code is embedded into an XML structure, the characters “<” and “>” cannot be used and must therefore be replaced with *&gt;* and *&lt;* when being used in the code.

The following shader code example also uses the replacements.

Example code “Media Overlay Add”



```
1   <?xml encoding="utf-8" ?>
2   <descripFx type = "fxShader" name = "Media Overlay Add" separator = "FX" rev = "2" forceLead = "false">
3       <descripInterface>
4           <descripModule type = "paramScalarValue" upper = "255" lower = "0" hlslType = "float" name = "Mix" hlslName = "mix" usage = "mixColorOutput" default = "0" artNetWidth = "1" normalize = "true"/>
5           <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "X" hlslName = "offX" default = "32768" artNetWidth = "2" normalize = "true" linkage = "open"/>
6           <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Y" hlslName = "offY" default = "32768" artNetWidth = "2" normalize = "true" linkage = "add"/>
7           <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Width" hlslName = "factorX" default = "65535" artNetWidth = "2" normalize = "true" linkage = "open"/>
8           <descripModule type = "paramScalarValue" upper = "65535" lower = "0" hlslType = "float" name = "Height" hlslName = "factorY" default = "65535" artNetWidth = "2" normalize = "true" linkage = "add"/>
9           <descripModule type = "paramResMedia" hlslType = "sampler2d" name = "Media" hlslName = "mediaTexSampler" artNetWidth = "2">
10              <descripSubModule hlslType = "float" hlslName = "texFormat" usage = "texFormat"/>
11              <descripSubModule hlslType = "float4" hlslName = "texMod" usage = "texCoordMod"/>
12              <descripSubModule hlslType = "bool" hlslName = "mpegColorSpace" usage = "mpegColorSpace"/>
13          </descripModule>
14      </descripInterface>
15      <descripShader type = "ps_5_0">
16          <pass>
17              <shaderCode>
18                  #include "SampleCorrected"
19
20                  float mix;
21                  float factorX;
22                  float factorY;
23                  float offX;
24                  float offY;
25                  float texFormat;
26                  float4 texMod;
27                  bool mpegColorSpace;
28
29                  Texture2D mediaTex;
30                  SamplerState mediaSampler;
31
32                  export void MediaOverlayMix(float4 col : COLOR0, float2 uv : TEXCOORD0, out float4 color : SV_Target0)
33                  {
34                      float2 factor = float2(factorX, factorY);
35
36                      float2 biasedTex = (uv * texMod.xy) / factor - 0.5 *(texMod.xy / factor);
37                      biasedTex.x += (1-(-3 + 7 * offX)) * texMod.x;
38                      biasedTex.y += (-3 + 7 * offY) * texMod.y;
39
40                      if(biasedTex.x &gt; 0 && biasedTex.x &lt; texMod.x && biasedTex.y &gt; 0 && biasedTex.y &lt; texMod.y)
41                      {
42                          float4 curcol = SampleCorrected(mediaTex, mediaSampler, biasedTex, mpegColorSpace, texFormat);
43                          curcol.rgb *= curcol.a;
44                          curcol += col;
45
46                          color = mix * curcol + (1.0 - mix) * col;
47                      }
48                      else
49                      {
50                          color = col;
51                      }
52                  }
53              </shaderCode>
54          </pass>
55      </descripShader>
56  </descripFx>
```

line 9:	*hlslName* must be set as *myVariableName**TexSampler*** in order to link **tex**ture and **sampler** which are separately declared in lines 29-30
lines 10-12:	must be included to receive information from a sampled *Pandoras Box* resource
line 18:	allows to use the *SampleCorrected* function within the shader code
lines 20-27:	declaration of variables used in the UI description
lines 29-30:	the variables must be matching the naming of line 9, but as separate variables
line 32:	each FX needs to have an export function in which the main program runs
line 42:	*SampleCorrected* allows for using different texture formats (compressed or uncompressed) within a single shader as it corrects the original format

The export function has an output of type float4 called *color* which is the shader’s return value.

The function also has 2 Input values. *uv* is the position of the currently processed pixel and *col* is its current color value.

Specific *Pandoras Box* variables

The following *Pandoras Box* variables can be used to make the most of your shaders:

Name	Type	Usage
pbProjectStartTime	float4	elapsed time since project start x: only to be used with below formula y: only to be used with below formula z: seconds w: milliseconds Max: 16776216 *elapsedTimeMs = x + y * Max*
pbCurrentTexDimension	float4	size of current DirectX texture in pixels x: width y: height
pbCurrentTexCoordMod	float4	factor for UV coordinates
pbMainMediaDimension	float4	size of main media in pixels x: width y: height
pbCurrentTex pbCurrentSampler	Texture2D SamplerState	Samples current texture including changes from previous effects. must be used with: - *„forceLead = true“* *pbCurrentTex.Sample(pbCurrentSampler, uv * pbCurrentTexCoordMod.xy)*
pbRandomTex pbRandomSampler	Texture2D SamplerState	samples a random texture *pbRandomTex.Sample(pbRandomSampler, uv)*
pbShaderPass	float	returns current render pass