Using Fragment Shaders to Manipulate Imagery

Mike Bailey
Oregon State University

Image Basics

Treat the image as a texture

To get from the current texel to a neighboring texel, add \((\text{ResS}^{-1}, \text{ResT}^{-1})\) to the current \((S, T)\)
Image Negative

( R, G, B )

( 1.-R, 1.-G, 1.-B )

uniform float S, T;
uniform float Power;
uniform sampler2D TexUnit;

void main()
{
vec2 st = gl_TexCoord[0].st;
vec2 delta = st - vec2(S, T);
st = vec2(S, T) + sign(delta) * pow(abs(delta), Power);
vec3 rgb = texture2D(TexUnit, st).rgb;
gl_FragColor = vec4(rgb, 1.);
}

Image Distortion
Image Un-Masking

What I don't want
More of what I do want
What I started with
What I don't want

Blend of what don't want and what have
Blend of what have and what want more of

\[ I_{out} = (1.-t)I_{don't want} + tI_{in} \]

Brightness

\[ I_{don't want} = \text{vec3}(0.,0.,0.); \]

T = 0.

T = 1.

T = 2.
Contrast

\[ Idontwant = \text{vec3}(0.5, 0.5, 0.5); \]

\[ T = 0. \quad T = 1. \quad T = 2. \]

HDTV Luminance Standard

\[ \text{Luminance} = 0.2125 \times \text{Red} + 0.7154 \times \text{Green} + 0.0721 \times \text{Blue} \]
**Saturation**

\[
\text{Idontwant} = \text{vec3}(\text{luminance, luminance, luminance});
\]

**Difference**

\[
\text{Idontwant} = I_{\text{before}}
\]
\[
I_{\text{in}} = I_{\text{after}}
\]
ChromaKey

Replace fragment if:
\[ R < T \]
\[ G < T \]
\[ B > 1. - T \]

T = 0.
T = 0.5
T = 1.

Blur

Blur Convolution:

\[
B = \frac{1.}{16.} \begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix}
\]
Sharpening

Blur Convolution:

\[ B = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ I_{\text{dontwant}} = I_{\text{blur}} \]
Embossing

\[
\begin{align*}
\text{vec2 stp0} &= \text{vec2}(1./\text{ResS}, 0.); \\
\text{vec2 stpp} &= \text{vec2}(1./\text{ResS}, 1./\text{ResT}); \\
\text{vec3 c00} &= \text{texture2D(ImageUnit, st).rgb}; \\
\text{vec3 cp1p1} &= \text{texture2D(ImageUnit, st + stpp).rgb}; \\
\text{vec3 diffs} &= \text{c00} \cdot \text{cp1p1}; \\
\text{float max} &= \text{diffs.r}; \\
\text{if( abs(diffs.g) > abs(max) )} & \\
\text{max} &= \text{diffs.g}; \\
\text{if( abs(diffs.b) > abs(max) )} & \\
\text{max} &= \text{diffs.b}; \\
\text{float gray} &= \text{clamp}(\text{max} + 5., 0., 1.); \\
\text{vec4 grayVersion} &= \text{vec4(gray, gray, gray, 1.);} \\
\text{vec4 colorVersion} &= \text{vec4(gray*c00, 1.);} \\
\text{color} &= \text{mix(grayVersion, colorVersion, T);}
\end{align*}
\]

Edge Detection

Horizontal and Vertical Sobel Convolutions:

\[
H = \begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix}
\quad
V = \begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}
\]

\[
S = \sqrt{H^2 + V^2}
\]

\[
\Theta = \text{atan2}(V, H)
\]
Edge Detection

vec2stp0 = vec2(1./ResS, 0.);
vec2stp0p = vec2(1./ResS, 1./ResT);
vec2stp0m = vec2(1./ResS, -1./ResT);
vec2stpmp = vec2(1./ResS, 0.);
vec2stpmpm = vec2(1./ResS, 1./ResT);
vec2stp1p = vec2(0., 1./ResT);
vec2stp1m = vec2(0., -1./ResT);
vec2stp10 = vec2(0., 0.);
vec2stp10p = vec2(0., 1./ResT);
vec2stp10m = vec2(0., -1./ResT);
vec2stp10mp = vec2(0., 0.);

float i00 = dot( texture2D( ImageUnit, st ).rgb, vec3(0.2125,0.7154,0.0721) );
float im1m1 = dot( texture2D( ImageUnit, st-stpp ).rgb, vec3(0.2125,0.7154,0.0721) );
float ip1p1 = dot( texture2D( ImageUnit, st+stpp ).rgb, vec3(0.2125,0.7154,0.0721) );
float im1p1 = dot( texture2D( ImageUnit, st-stpm ).rgb, vec3(0.2125,0.7154,0.0721) );
float ip1m1 = dot( texture2D( ImageUnit, st+stpm ).rgb, vec3(0.2125,0.7154,0.0721) );
float im10 = dot( texture2D( ImageUnit, st-stp0 ).rgb, vec3(0.2125,0.7154,0.0721) );
float ip10 = dot( texture2D( ImageUnit, st+stp0 ).rgb, vec3(0.2125,0.7154,0.0721) );
float i0m1 = dot( texture2D( ImageUnit, st-st0p ).rgb, vec3(0.2125,0.7154,0.0721) );
float i0p1 = dot( texture2D( ImageUnit, st+st0p ).rgb, vec3(0.2125,0.7154,0.0721) );

float h = -1.*im1p1 - 2.*i0p1 - 1.*ip1p1 + 1.*im1m1 + 2.*i0m1 + 1.*ip1m1;
float v = -1.*im1m1 - 2.*im10 - 1.*im1p1 + 1.*ip1m1 + 2.*ip10 + 1.*ip1p1;
float mag = sqrt( h*h + v*v );
vec3 target = vec3( mag, mag, mag );
color = vec4( mix( irgb, target, T ), 1. );

T = 0.
T = 0.5
T = 1.
Toon Rendering

```c
float mag = sqrt( h*h + v*v );
if( mag > MagTol )
{
    color = vec4( 0., 0., 0., 1. );
}
else
{
    rgb.rgb *= Quantize;
    rgb.rgb += vec3( .5, .5, .5 );
    ivec3 irgb = ivec3( rgb.rgb );
    rgb.rgb = vec3( irgb ) / Quantize;
    color = vec4( rgb, 1. );
}
```
Toon Rendering for Non-Photorealistic Effects

Use the GPU to enhance scientific, engineering, and architectural illustration

Use the GPU to enhance scientific, engineering, and architectural illustration
Toon Rendering for Non-Photorealistic Effects

Use the GPU to enhance scientific, engineering, and architectural illustration.

Mandelbrot Set

\[ Z_{i+1} = Z_i^2 + Z_0 \]

How fast does it converge, if ever?
Julia Set

\[ Z_{i+1} = Z_i^2 + C \]

How fast does it converge, if ever?