Using Fragment Shaders to Manipulate Images

Mike Bailey
mjb@cs.oregonstate.edu
Oregon State University

Image Basics

Treat the image as a texture. Index it using usual texture indexing (0 ≤ s, t ≤ 1.)

If you need it, the resolution of this texture can be found by saying:

ivec2 ires = textureSize(ImageUnit, 0);
float ResS = float(ires.s);
float ResT = float(ires.t);

To get from the current texel to a neighboring texel, add

±(1./ResS, 1./ResT)
to the current (S, T)

Image Negative

(R, G, B)

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

Image Distortion

uniform float uS0, uT0;
uniform float uPower;
uniform sampler2D uTexUnit;
in vec2 vST;

void main() {
vec2 delta = vST - vec2(uS0, uT0);
s = vec2(uS0, uT0) + sign(delta) * pow(abs(delta), uPower);
vec3 rgb = texture2D(uTexUnit, vST.rgb);
vec3 uRGB = vec3(uS0, uT0, uPower);
vec4 gl_FragColor = vec4(rgb, 1.0);
}
Image Un-masking:
Interpolation can still happen when $t < 0.$ or $t > 1.$

$$Q = (1-t)Q_0 + tQ_1$$

$t = -1.$

$t = 0.$

$t = 1.$

$t = 2.$

Abusing the Linear Blending Equation for a Good Purpose

$$I_{\text{out}} = (1-t)I_{\text{dontwant}} + tI_{\text{in}}$$

$0.0$ $1.0$ $2.0$

$t$

More of what I do want

Blend of what I have and less of what I don’t want

What I have to start with

What I don’t want

Blend of what I have and what I don’t want

Brightness

$$I_{\text{dontwant}} = \text{vec3}(0., 0., 0.);$$

Contrast

$$I_{\text{dontwant}} = \text{vec3}(0.5, 0.5, 0.5);$$
HDTV Luminance Standard

Luminance = 0.2125*Red + 0.7154*Green + 0.0721*Blue

Saturation

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

\[ T = 0. \quad T = 1. \quad T = 3. \]

Difference

\[ \text{Idontwant} = \text{Ibefore} \]
\[ \text{In} = \text{Iafter} \]

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

ChromaKey

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

\[ T = 0. \quad T = 0.5 \quad T = 1. \]
Blur Convolution:

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

Sharpening

\[
\text{I}_{\text{want}} = \text{I}_{\text{blur}}
\]

\[
\text{vec2 stp0} = \text{vec2}(1./\text{ResS}, 0., 0.);
\text{vec2 st0p} = \text{vec2}(0., 1./\text{ResT});
\text{vec2 stpp} = \text{vec2}(1./\text{ResS}, 1./\text{ResT});
\text{vec2 stpm} = \text{vec2}(1./\text{ResS}, -1./\text{ResT});
\text{vec3 I00} = \text{texture2D(uImageUnit, vST).rgb};
\text{vec3 I1m1} = \text{texture2D(uImageUnit, vST-stpp).rgb};
\text{vec3 I1p1} = \text{texture2D(uImageUnit, vST+stpp).rgb};
\text{vec3 I1p1} = \text{texture2D(uImageUnit, vST-stpm).rgb};
\text{vec3 I1m1} = \text{texture2D(uImageUnit, vST+stpm).rgb};
\text{vec3 I0m1} = \text{texture2D(uImageUnit, vST-stp0).rgb};
\text{vec3 I0p1} = \text{texture2D(uImageUnit, vST+stp0).rgb};
\text{vec3 I00} = \text{texture2D(uImageUnit, vST).rgb};
\text{vec3 target} = \text{vec3}(0.0, 0.0, 0.0);
\text{target} += 1.0i(\text{I1m1}+\text{I1p1}+\text{I1p1}+\text{I1m1});
\text{target} += 2.0i(I00+I0m1+I0p1);
\text{target} += 4.0i(I00); 
\text{target} /= 16.0;
\text{gl_FragColor} = \text{vec4}\text{mix(target, irgb, T), 1.1};
Embossing

```glsl
vec2 stp0 = vec2(1./ResS, 0.);
vec2 stpp = vec2(1./ResS, 1./ResT);
vec3 c00 = texture2D(uImageUnit, vST).rgb;
vec3 cp1p1 = texture2D(uImageUnit, vST + stpp).rgb;
dvec2 diffs = c00 - cp1p1;
float max = diffs.r;
if( abs(diffs.g) > abs(max) )
    max = diffs.g;
if( abs(diffs.b) > abs(max) )
    max = diffs.b;
float gray = clamp( max + .5, 0., 1.);
vec4 grayVersion = vec4(gray, gray, gray, 1.);
vec4 colorVersion = vec4(gray * c00, 1.);
gl_FragColor = mix(grayVersion, colorVersion, T);
```

Horizontal and Vertical Sobel Convolutions:

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

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

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

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

Edge Detection

```glsl
const vec3 LUMCOEFFS = vec3(0.2125, 0.7154, 0.0721);
vec2 stp0 = vec2(1./ResS, 0.);
vec2 stpp = vec2(1./ResS, 1./ResT);
float i00 = dot(texture2D(uImageUnit, vST).rgb, LUMCOEFFS);
float im1m1 = dot(texture2D(uImageUnit, vST-stpp).rgb, LUMCOEFFS);
float ip1p1 = dot(texture2D(uImageUnit, vST+stpp).rgb, LUMCOEFFS);
float im1p1 = dot(texture2D(uImageUnit, vST-stpm).rgb, LUMCOEFFS);
float ip1m1 = dot(texture2D(uImageUnit, vST+stpm).rgb, LUMCOEFFS);
float im10 = dot(texture2D(uImageUnit, vST-stp0).rgb, LUMCOEFFS);
float ip10 = dot(texture2D(uImageUnit, vST+stp0).rgb, LUMCOEFFS);
float i0m1 = dot(texture2D(uImageUnit, vST-st0p).rgb, LUMCOEFFS);
float i0p1 = dot(texture2D(uImageUnit, vST+st0p).rgb, LUMCOEFFS);

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(grayVersion, colorVersion, T).rgb, 1.);
```
float mag = sqrt(h*h + v*v);
if( mag > uMagTol )
{
    gl_FragColor = vec4( 0., 0., 0., 1.);
}
else
{
    rgb.rgb *= uQuantize;
    rgb.rgb += vec3( .5, .5, .5 );
    ivec3 irgb = ivec3( rgb.rgb );
    rgb.rgb = vec3( irgb ) / uQuantize;
    gl_FragColor = vec4( rgb, 1. );
}

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?

Using Double Precision

Can Do Image Processing on Dynamic Scenes with a Two-pass Approach

Pass #1

- Render a 3D dynamic scene
- Lighting Shader
- Texture

Pass #2

- Render a quadrilateral
- Blur Shader
- Framebuffer