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:
vec2 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);
st = vec2(uS0,uT0) + sign(delta) * pow( abs(delta), uPower);
vec3 rgb = texture2D( uTexUnit, vST ).rgb;
gl_FragColor= vec4( rgb, 1. );
}

Image Un-masking:
Interpolation can still happen when t < 0. or t > 1.

Image Un-Masking:
Abusing the Linear Blending Equation for a Good Purpose

\[ Q = (1-t)Q_0 + tQ_1 \]

Where

- \( t \) is the time parameter
- \( Q_0 \) and \( Q_1 \) are the initial and final states

More of what I do want

What I have to start with

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

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

What I don't want

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

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

\[ \text{Idontwant} = \text{vec3}(0., 0., 0.); \]

Contrast

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

HDTV Luminance Standard

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

Saturation

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

Difference

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

ChromaKey

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

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

Sharpening

\[ \text{Idontwant} = \text{Iblur} \]

Embossing

\[ \text{diffg} = \text{diffb} = \text{sift}; \]

\[ \text{diffa} = \text{sift}; \]

\[ \text{targets} = \text{sift}; \]

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

const vec3 LUMCOEFFS = vec3(0.2125, 0.7154, 0.0721);
vec2 stp0 = vec2(1./ResS, 0.);
vec2 st0p = vec2(0., 1./ResT);
vec2 stpp = vec2(1./ResS, 1./ResT);
vec2 stpm = 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 im0 = 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(irgb, target, T), 1.);

Toon Rendering

Toon Rendering for Non-Photorealistic Effects

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

Toon Rendering for Non-Photorealistic Effects
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
- Texture

Pass #2

- Render a quadrilateral
- Framebuffer
  - Lighting Shader
  - Blur Shader