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

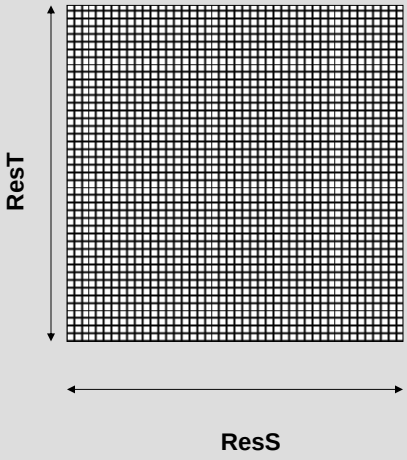


Image Negative

(R, G, B)

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

Image Distortion

```
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. );
}
```

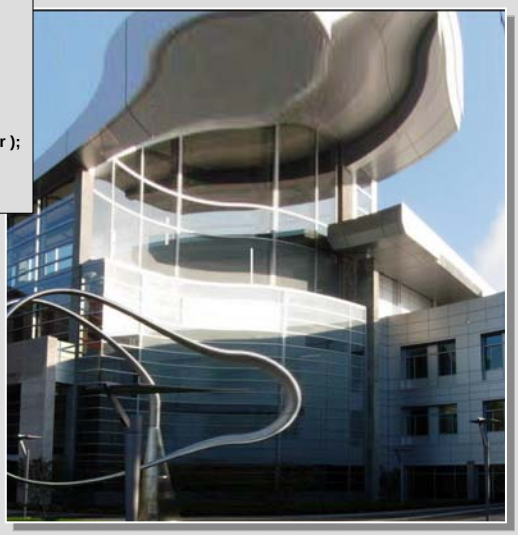


Image Un-Masking

$$I_{out} = (1.-t)*I_{dontwant} + t*I_{in}$$

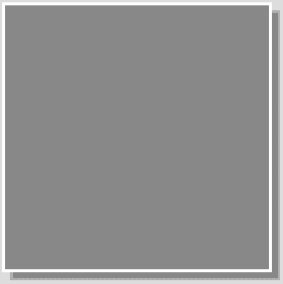
Brightness

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

T = 0. T = 1. T = 2.

Contrast

$I_{dontwant}$ = vec3(0.5, 0.5, 0.5);



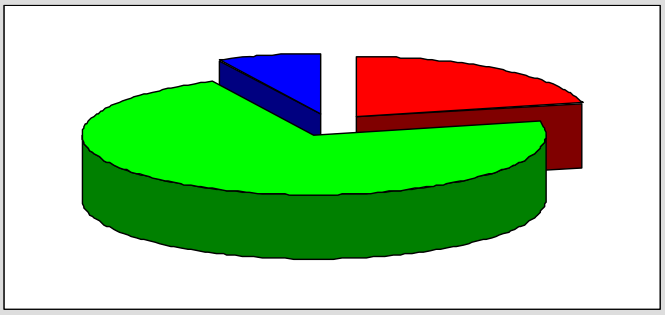
T = 0.

T = 1.

T = 2.

HDTV Luminance Standard

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



Saturation

$I_{dontwant}$ = vec3(luminance, luminance, luminance);

T = 0.

T = 1.

T = 3.

Difference

$I_{dontwant} = I_{before}$

$I_{in} = I_{after}$

T = 0.

T = 1.

T = 2.

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:

$$B = \frac{1.}{16.}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$I_{dontwant} = I_{blur}$$

Sharpening

T = 0.

T = 1.

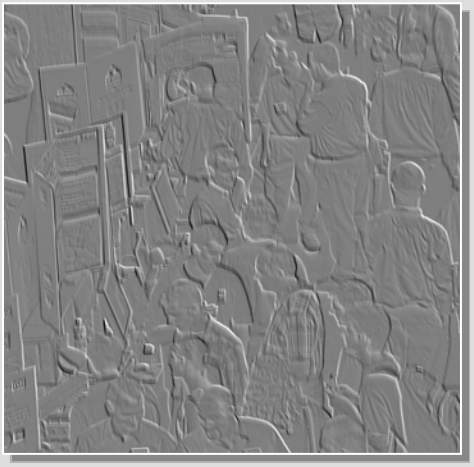
T = 2.

Embossing

```
vec2 stp0 = vec2( 1./ResS,  0. );
vec2 stpp = vec2( 1./ResS,  1./ResT);
vec3 c00    = texture2D( ImageUnit, st ).rgb;
vec3 cp1p1  = texture2D( ImageUnit, st + stpp ).rgb;

vec3 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. );
color = mix( grayVersion, colorVersion, T );
```



Edge Detection

Horizontal and Vertical Sobel Convolutions:

$$H = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

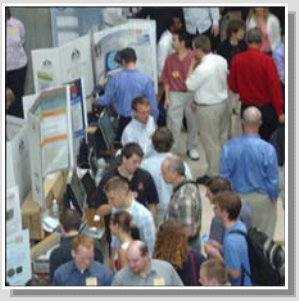
$$S = \sqrt{H^2 + V^2} \qquad \Theta = \text{atan2}(V, H)$$

Edge Detection

```
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( 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. );
```


Edge Detection



T = 0.

T = 0.5

T = 1.

Toon Rendering

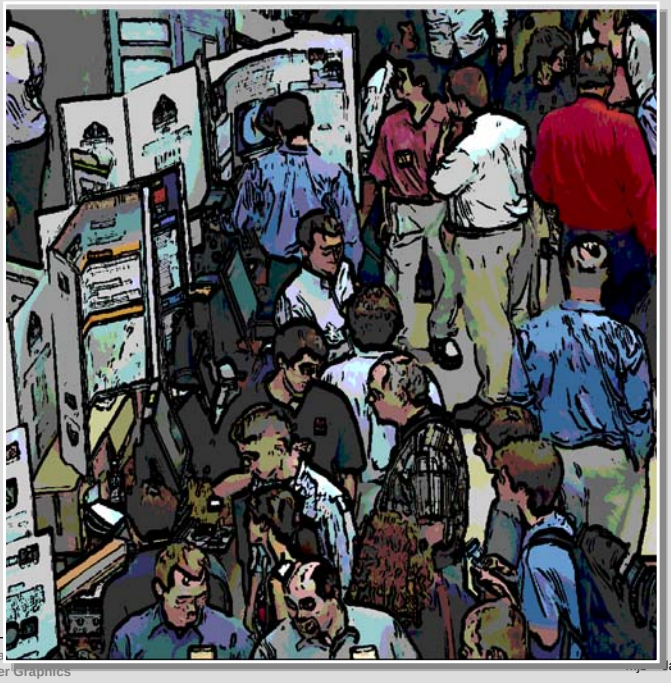
```
float mag = sqrt( h*h + v*v );
if( mag > MagTol )
{
        color = vec4( 0., 0., 0., 1. );
}
else
{
        rgb.rgb *= Quantize;
        rgb.rgb += vec3( .5, .5, .5 );
        ivec3 irgb = ivec3( rgb.rgb );
        rgb.rgb = vec3( irgb ) / Quantize;
        color = vec4( rgb, 1. );
}
```


Toon Rendering

Original Image

Colors Quantized

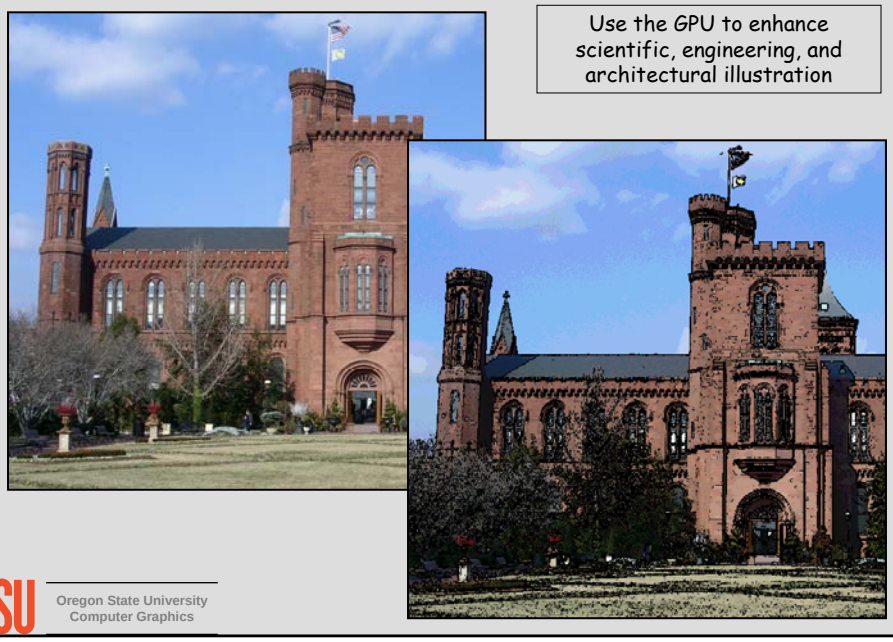
Outlines Added



Toon Rendering for Non-Photorealistic Effects

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

Toon Rendering for Non-Photorealistic Effects

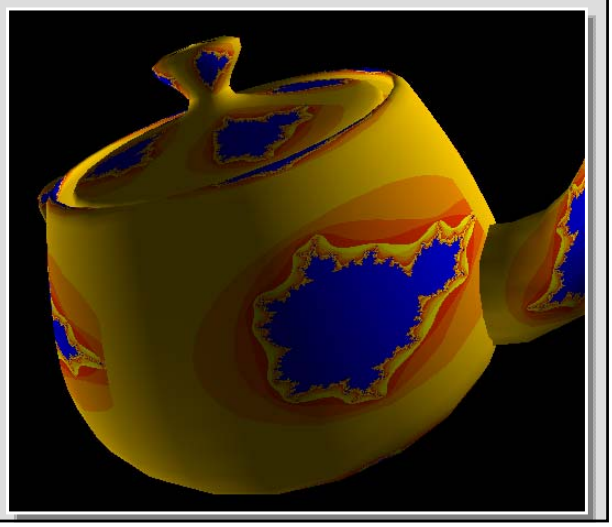


Use the GPU to enhance scientific, engineering, and architectural illustration

Mandelbrot Set

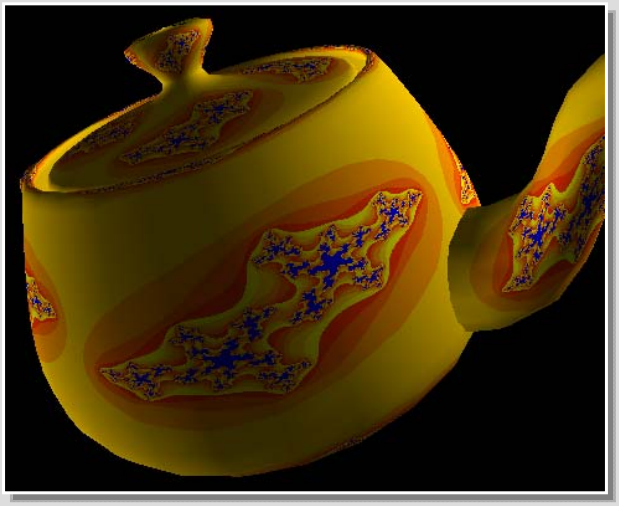
$$Z_{i+1} = Z_i^2 + Z_0$$

How fast does it converge, if ever?



Julia Set

$$Z_{i+1} = Z_i^2 + C$$



How fast does it converge, if ever?