Project 5

Alan Neads

27th February, 2017

mag.glib

```
##OpenGL GLIB
Ortho -5. 5.   -5. 5.
LookAt 0 0 2  0 0 0  0 1 0

Texture2D 5 image.bmp

Vertex   mag.vert
Fragment mag.frag
Program  Mag                                            \
        uScenter <0. .5 1.>                             \
        uTcenter <0. .5 1.>                             \
        uDs      <0.01 .1 .5>                      \
        uDt      <0.01 .1 .5>                      \
        uCircle  <0>                                    \
        uCircleDiam <0.01 0.1 0.5>                 \
        uMagFactor <0.0 0.5 1.0>                   \
        uRotAngle <-3.14159 0. 3.14159>            \
        uSharpFactor <0. 1. 5.>                         \
        uImageUnit  5                                   \
        uResS 1024                                           \
        uResT 1024

QuadXY .2 5.

MessageBox Circle magnification is implemented
```

mag.vert

```
void main( void )
{
	gl_TexCoord[0] = gl_MultiTexCoord0;
	gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

decaying.frag

```
uniform sampler2D uImageUnit;

uniform float uScenter;
uniform float uTcenter;
uniform float uDs;
uniform float uDt;
uniform bool uCircle;
uniform float uCircleDiam;
uniform float uMagFactor;
uniform float uRotAngle;
uniform float uSharpFactor;
uniform float uResS;
uniform float uResT;

vec2 rotate_point( vec2 pivot, vec2 pos, float theta )
{
        vec2 output = vec2( pos.s - pivot.s, pos.t - pivot.t );
        float s = sin(theta);
        float c = cos(theta);
        float rotx = output.s * c - output.t * s;
        float roty = output.s * s + output.t * c;
        output.s = output.s + pivot.s + rotx;
        output.t = output.t + pivot.t + roty;
        return output;
}

vec3 sharpen( vec2 st, vec3 irgb, float ResS, float ResT, sampler2D ImageUnit, float T )
{
        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);
        vec3 i00 =   texture2D( ImageUnit, st ).rgb;
        vec3 im1m1 = texture2D( ImageUnit, st-stpp ).rgb;
        vec3 ip1p1 = texture2D( ImageUnit, st+stpp ).rgb;
        vec3 im1p1 = texture2D( ImageUnit, st-stpm ).rgb;
        vec3 ip1m1 = texture2D( ImageUnit, st+stpm ).rgb;
        vec3 im10 =  texture2D( ImageUnit, st-stp0 ).rgb;
        vec3 ip10 =  texture2D( ImageUnit, st+stp0 ).rgb;
        vec3 i0m1 =  texture2D( ImageUnit, st-st0p ).rgb;
        vec3 i0p1 =  texture2D( ImageUnit, st+st0p ).rgb;
        vec3 target = vec3(0.,0.,0.);
        target += 1.*(im1m1+ip1m1+ip1p1+im1p1);
        target += 2.*(im10+ip10+i0m1+i0p1);
```

```
	target += 4.*(i00);
	target /= 16.;
	return vec4( mix( target, irgb, T ), 1. ).rgb;
}
void main( void )
{
	vec2 st = gl_TexCoord[0].st;
	vec3 tex = texture2D( uImageUnit, st ).rgb;

	if ( uCircle ) {
		float dist = (st.s - uScenter) * (st.s - uScenter) + (st.t - uTcenter) * (st.t - uTcenter);
		dist = sqrt(dist);
		if ( dist <= uCircleDiam ) {
			vec2 magst = vec2( st.s, st.t );
			magst.s = magst.s + ((uScenter - magst.s) * uMagFactor);
			magst.t = magst.t + ((uTcenter - magst.t) * uMagFactor);
			vec2 rotst = rotate_point( vec2(uScenter, uTcenter), magst, uRotAngle );
			tex = texture2D( uImageUnit, rotst ).rgb;
			tex = sharpen( rotst, tex, uResS, uResT, uImageUnit, uSharpFactor );
		}
	} else {
		if ( st.s >= uScenter - uDs && st.s <= uScenter + uDs ) {
			if ( st.t >= uTcenter - uDt && st.t <= uTcenter + uDt ) {
				vec2 magst = vec2( st.s, st.t );
				magst.s = magst.s + ((uScenter - magst.s) * uMagFactor);
				magst.t = magst.t + ((uTcenter - magst.t) * uMagFactor);
				vec2 rotst = rotate_point( vec2(uScenter, uTcenter), magst,
uRotAngle );
				tex = texture2D( uImageUnit, rotst ).rgb;
				tex = sharpen( rotst, tex, uResS, uResT, uImageUnit, uSharpFactor
);
			}
		}
	}

	gl_FragColor.rgb = tex;
	gl_FragColor.a = 1.0;
}
```

Deconstruction

I started with a simple vertex shader that just reports the texture coordinates to the fragment shader. Then I performed a texture lookup with the fragments' coordinates. This draws the basic image specified within the glib file to the surface. To implement the magnification we determine if the current fragment falls within the square zone, or optionally calculate a distance from the center and compare that to a radius value of the circle zone. Each of these operations have been done before in prior assignments. To magnify the zone we determine how far the fragment is from the center of the zone, then scale the distance by the magnification value. Thus the higher this value goes, the more zooming factor there will be, and if we approach zero we end up zooming out. To implement rotation, we simply need to compute the difference of position from the center, then rotate that point around an imaginary origin of (0,0), then we rotate each axis, then add the resulting value to the original position. This allows us to rotate the image, however it does suffer from zooming in as you approach positive or negative pi, I assume this is an example case of gimbal lock. Lastly to implement sharpening I utilized the formula provided in the handouts section, in which it's just amplifying the current fragment, by determining the effects and edges from adjacent pixels within the image. If the current fragment doesn't fall within either the square zone or circle zone, we simply use the images' pixel at the screen position.

Images

M GI 8830

M GI 8830

M GI 8830

M GI 8830

M GI 8830

M GI 8830

M GI 8830