CS557 Project #5

Yunfan Li

[liyunf@oregonstate.edu](mailto:liyunf@oregonstate.edu)

1. **Source Listings**
   1. ***liyunf.glib***

##OpenGL GLIB

Ortho -5. 5. -5. 5.

LookAt 0 0 2 0 0 0 0 1 0

Texture2D 5 image.bmp

Vertex liyunf.vert

Fragment liyunf.frag

Program Mag \

uScenter <0. .5 1.> \

uTcenter <0. .5 1.> \

uDs <0.01 .1 .5> \

uDt <0.01 .1 .5> \

uMagFactor <.1 1. 25.> \

uRotAngle <-3.14159 0. 3.14159> \

uSharpFactor <0. 1. 5.> \

uImageUnit 5

QuadXY .2 5.

* 1. ***liyunf.vert***

#version 330 compatibility

out vec2 vST;

void

main( )

{

vST = gl\_MultiTexCoord0.st;

gl\_Position = gl\_ModelViewProjectionMatrix \* gl\_Vertex;

}

* 1. ***liyunf.frag***

#version 330 compatibility

in vec2 vST;

uniform float uScenter;

uniform float uTcenter;

uniform float uDs;

uniform float uDt;

uniform float uMagFactor;

uniform float uRotAngle;

uniform float uSharpFactor;

uniform sampler2D uImageUnit;

void

main ()

{

vec2 ct = vec2(uScenter, uTcenter);

vec3 rgb = texture2D(uImageUnit, vST).rgb;

if ((vST.s>(ct.s-uDs)) && (vST.s<(ct.s+uDs)) && (vST.t>(ct.t-uDt)) && (vST.t<(ct.t+uDt))) {

vec2 magst = (vST - ct)/uMagFactor;

vec2 rotst = vec2(magst.s\*cos(uRotAngle)-magst.t\*sin(uRotAngle), magst.s\*sin(uRotAngle)+magst.t\*cos(uRotAngle));

vec2 comst = rotst + ct;

ivec2 ires = textureSize(uImageUnit, 0);

vec3 irgb = texture2D(uImageUnit, comst).rgb;

float ResS = float(ires.s);

float ResT = float(ires.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( uImageUnit, comst ).rgb;

vec3 im1m1 = texture2D( uImageUnit, comst-stpp ).rgb;

vec3 ip1p1 = texture2D( uImageUnit, comst+stpp ).rgb;

vec3 im1p1 = texture2D( uImageUnit, comst-stpm ).rgb;

vec3 ip1m1 = texture2D( uImageUnit, comst+stpm ).rgb;

vec3 im10 = texture2D( uImageUnit, comst-stp0 ).rgb;

vec3 ip10 = texture2D( uImageUnit, comst+stp0 ).rgb;

vec3 i0m1 = texture2D( uImageUnit, comst-st0p ).rgb;

vec3 i0p1 = texture2D( uImageUnit, comst+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.;

rgb = mix( target, irgb, uSharpFactor );

}

gl\_FragColor = vec4(rgb, 1.);

}

1. **What I did & Why it worked**

***2.1 What I did***

In my liyunf.glib file, I created a Quad extended in XY plane and connect this .glib flie with liyunf.vert vertex shader and liyunf.frag fragment shader. Assign one BMP file as my textrure file and put it in texture unit in GPU.

In my liyunf.vert file, I get the s and t coordinates for further using and generate gl\_Position variable.

In my liyunf.frag flie, I calculate original image resolution and then check if this fragment is in our magic lens. If it is, I will “magnify” this fragment at first, my approach of this fake magnify is I calculate a vector which points from center to this fragment and then divided by uMagFactor to get a new short vector but points to the same direction, then add this new vector to center coordinates to get a new fake st coordinates for this fragment. After magnify this fragment, I rotate it based on equations on project websites. At last, I used sample code in slides to implement sharpen effect based on rotated st coordinates.

***2.2 Why it worked***

Fake st coordinates are generated by center to current fragment vector, this vector will give us a correct direction but after divided by magnify factor, the vector become shorter than original one. Means we can get a pixel closer to center, and display this pixel in a further point, by which we get the magnify effects.

1. **Images**

****

****

****

****