CS 557 -- Winter Quarter 2016

Project #5 Report

Image Manipulation in a "Magic Lens"

Name: Kai Shi

ID: 932-504-512

In project5, I created a Magic Lens for an image display. Inside the Magic Lens, the image can be able to be magnified, rotated, and sharpened.

(1) Code

glib:

```
##OpenGL GLIB
Ortho -5. 5.   -5. 5.
LookAt 0 0 2  0 0 0  0 1 0
Texture2D 5 minions.bmp
Vertex   kaishi.vert
Fragment kaishi.frag
Program kaishi                                                  \
        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.>                 \
        uCircle <false>                                 \
        uImageUnit  5
QuadXY .2 5.
```

vert:

```
#version 330 compatibility
out vec2 vST;
void main()
{
  vST = gl_MultiTexCoord0.st;
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

frag:

```
#version 330 compatibility
```

```
uniform sampler2D uImageUnit;
uniform float uScenter, uTcenter, uDs, uDt;
uniform float uMagFactor,uRotAngle;
uniform float uSharpFactor;
uniform bool uCircle;
in vec2 vST;
void main()
{
  vec2 st = vST;
  vec4 refractcolor;
  vec3 irgb;
  vec4 color = vec4( 1., 0., 0., 1. );
  if(uCircle)
  {
    if(((st.s-uScenter)*(st.s-uScenter)+(st.t-uTcenter)*(st.t-
uTcenter))<(uDs*uDs+uDt*uDt))
  {      //Mag
      st -= vec2(uScenter,uTcenter);
      st /= uMagFactor;

      //rotation
      float x_new = st.s*cos(uRotAngle) - st.t*sin(uRotAngle);
      float y_new = st.s*sin(uRotAngle) + st.t*cos(uRotAngle);
      st = vec2(x_new,y_new);
      st += vec2(uScenter,uTcenter);

      //Sharpning
      vec2 isize = textureSize( uImageUnit, 0 );
      float ResS = float( isize.s );
      float ResT = float( isize.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, st ).rgb;
      vec3 im1m1 = texture2D( uImageUnit, st-stpp ).rgb;
      vec3 ip1p1 = texture2D( uImageUnit, st+stpp ).rgb;
      vec3 im1p1 = texture2D( uImageUnit, st-stpm ).rgb;
      vec3 ip1m1 = texture2D( uImageUnit, st+stpm ).rgb;
      vec3 im10 =  texture2D( uImageUnit, st-stp0 ).rgb;
      vec3 ip10 =  texture2D( uImageUnit, st+stp0 ).rgb;
```

```
    vec3 i0m1 =  texture2D( uImageUnit, st-st0p ).rgb;
    vec3 i0p1 =  texture2D( uImageUnit, 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.;
    refractcolor = vec4(texture(uImageUnit,st).rgb,1.);
    irgb = texture( uImageUnit,  st ).rgb;
    color = vec4( mix( target, irgb, uSharpFactor ), 1. );
    refractcolor = color;
  }
  else
    refractcolor = vec4(texture2D( uImageUnit, st ).rgb, 1. );
  }
else{
if((abs(st.s-uScenter)<uDs/2.0)&&(abs(st.t-uTcenter)<uDt/2.0))
  {
    //Mag
    st -= vec2(uScenter,uTcenter);
    st /= uMagFactor;

    //rotation
    float x_new = st.s*cos(uRotAngle) - st.t*sin(uRotAngle);
    float y_new = st.s*sin(uRotAngle) + st.t*cos(uRotAngle);
    st = vec2(x_new,y_new);
    st += vec2(uScenter,uTcenter);

    //Sharpning
    vec2 isize = textureSize( uImageUnit, 0 );
    float ResS = float( isize.s );
    float ResT = float( isize.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, st ).rgb;
    vec3 im1m1 = texture2D( uImageUnit, st-stpp ).rgb;
    vec3 ip1p1 = texture2D( uImageUnit, st+stpp ).rgb;
    vec3 im1p1 = texture2D( uImageUnit, st-stpm ).rgb;
```

```
    vec3 ip1m1 = texture2D( uImageUnit, st+stpm ).rgb;
    vec3 im10 =  texture2D( uImageUnit, st-stp0 ).rgb;
    vec3 ip10 =  texture2D( uImageUnit, st+stp0 ).rgb;
    vec3 i0m1 =  texture2D( uImageUnit, st-st0p ).rgb;
    vec3 i0p1 =  texture2D( uImageUnit, 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.;
    refractcolor = vec4(texture(uImageUnit,st).rgb,1.);
    irgb = texture( uImageUnit,  st ).rgb;
    color = vec4( mix( target, irgb, uSharpFactor ), 1. );
    refractcolor = color;
  }
  else
   refractcolor = vec4(texture2D( uImageUnit, st ).rgb, 1. );
  }
   gl_FragColor = refractcolor;
}
```

(2) What I did and Reasons

First step I need to do is drawing a QuadXY and treat the bmp file as a texture. I defined the st coordinates as vST in the vert shader and passed it to the frag shader. In frag shader, define a st = vST, and defined a refractcolor = vec4(texture2D(uImageUnit, st).rgb, 1.), then, passed it to gl_FragColor. Unitl now, we can draw the original picture.

Next step is creating rectangle. If -uDs/2 <=current point.s – center.s<= uDs/2, and -uDt/2 <=current point.t – center.t<= uDt/2, it means current point is in the rectangle. So, in project, I just use abs() function to compare current point.s – center.s and uDs/2. Also does the uDt/2. Until now I have created a rectangle.

Next is doing magnification and rotation. Before doing this, to make sure the changes in rectangle are based on the center of the rectangle, I used st -= vec2(uScenter,uTcenter) to keep it. For magnification, I used st /= uMagFactor to create a new st. And for rotation, I just used float x_new = st.s*cos(uRotAngle) - st.t*sin(uRotAngle) and float y_new = st.s*sin(uRotAngle) + st.t*cos(uRotAngle) to create the new st and after doing this, I added center of rectangle.

Last step is doing sharpning. It is similar to the code in the notes. In the end, I set the refractcolor = vec4(mix(target, irgb, uSharpFactor), 1.).

Extra credit part, most codes are similar to the rectangle. Only thing need to be changed is the condition to judge whether current point is in the lens. If $(st.s-uScenter)^2 + (st.t-uTcenter)^2 <= radius$, it means the current point is in the circle lens.

(3) Results

The original:

Magnification:

Rotation:


C:\Users\Administrator\Desktop\HW5\kaishi.glib
SUPERB
WALLPAPERS
#1
DAD
WORLD'S
BEST

Sharpning (uSharpFactor = 0):

Sharpning (uSharpFactor = 5):

Changing lens location and size:

Extal Results:

Magnification:

Rotation:


C:\Users\Administrator\Desktop\HW5\kaishi.glib
SUPERB
WALLPAPERS
#1
DAD
WORLD'S
BEST
DAD

Sharpning (uSharpFactor = 0):

Changing lens location and size: