

RenderMan and OpenGL Shaders

CS557

Project # 6

Chao Zhang

1. Source listings

`##OpenGL GLIB`

`LookAt 0 0 7 0 0 0 0 1 0`

`Perspective 70`

`Vertex chaozhang.vert`

`Fragment chaozhang.frag`

`Program OvalNoise`

`Ad <.01 1. .5> Bd <.01 .1 .5>`

`NoiseAmp <0. 10. 10.> NoiseFreq <0. 1. 10.>`

`Alpha <0. 1. 1.> mColor {1., .5, 0. }`

`Tol <0. 0. 1.>`

`UseChromaDepth <false> \`

`ChromaRed <-10. -5. -1.> ChromaBlue <-10. -3. -1.> \`

`Obj deer.obj`

`chaozhang.glib`

This file include all the definition I need to use. The ChromaRed and ChromaBlue are both in the range of -10 to -1, and the default is -5. mColor is still the beaver orange. This time I didn't use the teal as background color because it is not clear to see the tol. All those will implement on the object deer.

`#version 330 compatibility`

`out vec3 vMCposition;`

`out vec4 vColor;`

`out float vLightIntensity;`

`out vec2 vST;`

`out float z;`

`const vec3 LIGHTPOS = vec3(-2., 0., 10.);`

`void`

`main()`

`{`

`vec3 tnorm = normalize(gl_NormalMatrix * gl_Normal);`

`vec3 ECposition = vec3(gl_ModelViewMatrix * gl_Vertex).xyz;`

`vLightIntensity = abs(dot(normalize(LIGHTPOS - ECposition), tnorm));`

`z = ECposition.z;`

`vMCposition = gl_Vertex.xyz;`

`vColor = gl_Color;`

`vST = gl_MultiTexCoord0.st;`

`gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;`

`}`

`chaozhang.vert`

Z= ECposition.z is the vertex position in eye coordinates. The vLightIntensity is the computed lighting in the vertex shader. tnorm for storing the transformed normal. The vMCposition, vcolor, vLightIntensity, vST and z will be trans to the .frag file.

```

#version 330 compatibility

in vec3  vMCposition;
in vec4  vColor;
in float vLightIntensity;
in vec2  vST;
in float z;

uniform bool  UseDiscard;
uniform float Ad;
uniform float Bd;
uniform vec4  mColor;
uniform float NoiseAmp;
uniform float NoiseFreq;
uniform sampler3D Noise3;
uniform float Tol;
uniform float Alpha;
uniform bool  UseChromaDepth;
uniform float ChromaRed;
uniform float ChromaBlue;

```

Part1 chaozhang.freq

This is part of the .freq file. This part will get the data from the vertex shader and define the variable.

```

ChromaDepth( float t )
{
    t = clamp( t, 0., 1. );

    float r = 1.;
    float g = 0.0;
    float b = 1. - 6. * ( t - (5./6.) );

    if( t <= (5./6.) )
    {
        r = 6. * ( t - (4./6.) );
        g = 0.;
        b = 1.;
    }

    if( t <= (4./6.) )
    {
        r = 0.;
        g = 1. - 6. * ( t - (3./6.) );
        b = 1.;
    }

    if( t <= (3./6.) )
    {
        r = 0.;
        g = 1.;
        b = 6. * ( t - (2./6.) );
    }

    if( t <= (2./6.) )
    {
        r = 1. - 6. * ( t - (1./6.) );
        g = 1.;
        b = 0.;
    }

    if( t <= (1./6.) )
    {
        r = 1.;
        g = 6. * t;
    }

    return vec3( r, g, b );
}

```

Part2 chanzhang.freq

This part is to implement the different color for the ChromaDepth. Use different color for different range of t. This can change the sphere into rainbow sphere.

```

void
main( )
{
    vec4 nv = texture3D( Noise3, NoiseFreq * vMCposition );
    float n = nv.r + nv.g + nv.b + nv.a;    // 1. -> 3.
    n = n - 2.;                            // -1. -> 1.
    float delta = NoiseAmp * n;
    float s = vST.s;
    float t = vST.t;
    float up = 2. * s;
    float vp = t;
    up += delta;
    vp += delta;
    int numins = int( up / Ad);
    int numint = int( vp / Bd);
    float Ar = Ad/2;
    float Br = Bd/2;

    gl_FragColor = vColor;    // default color
}

```

Part3 chaozhang.freq

This part use the Noise3 to create the moise. There are four values in this, the r,g,b and a. The “noise vector” texture nv is a vec4 whose components have separate meanings.

```
float n = nv.r + nv.g + nv.b + nv.a;
```

```
n = n - 2;
```

Those two lines made the range of the four-octave function from 0 to 1. The rest of the code is like the project1 and 2. It is for the eclipse.

```

gl_FragColor = vColor;          // default color

if( ( numins+numint ) % 1 ) == 0. )
{
    float uc = numins*2. * Ar + Ar;
    float vc = numint*2. * Br + Br;
    up = (up - uc)/Ar;
    vp = (vp - vc)/Br;
    float d = up * up + vp * vp;
    if( abs(d - 1) <= Tol )
    {
        float m = smoothstep( 1 - Tol, 1 + Tol, d);
        gl_FragColor = mix( mColor, vColor, m);
    }
    if( d <= 1.-Tol )
    {
        if( UseChromaDepth )
        {
            float t = (2./3.) * ( z - ChromaRed ) / ( ChromaBlue - ChromaRed );
            t = clamp( t, 0., 2./3. );
            gl_FragColor = vec4( ChromaDepth( t ) , 1. );
        }
        else
        {
            gl_FragColor = mColor;
        }
    }
    if( d >= 1.+Tol )
    {
        if( UseDiscard )
        {
            discard;
        }
        else
        {
            gl_FragColor = vec4( 1, 1, 0, Alpha);
        }
    }
}

gl_FragColor.rgb *= vLightIntensity;    // apply lighting model

```

part4 chaozhang.freq

This part is the most important part of this project.

```

if( abs(d - 1) <= Tol )
{
    float m = smoothstep( 1 - Tol, 1 + Tol, d);

    gl_FragColor = mix( mColor, vColor, m);
}

```

Those lines for the Tol function. With the increase of tol, the edge of the eclipse smooth. I mix the color to achieve the smooth goal.

```

if( d <= 1.-Tol )
{
    if( UseChromaDepth )

```

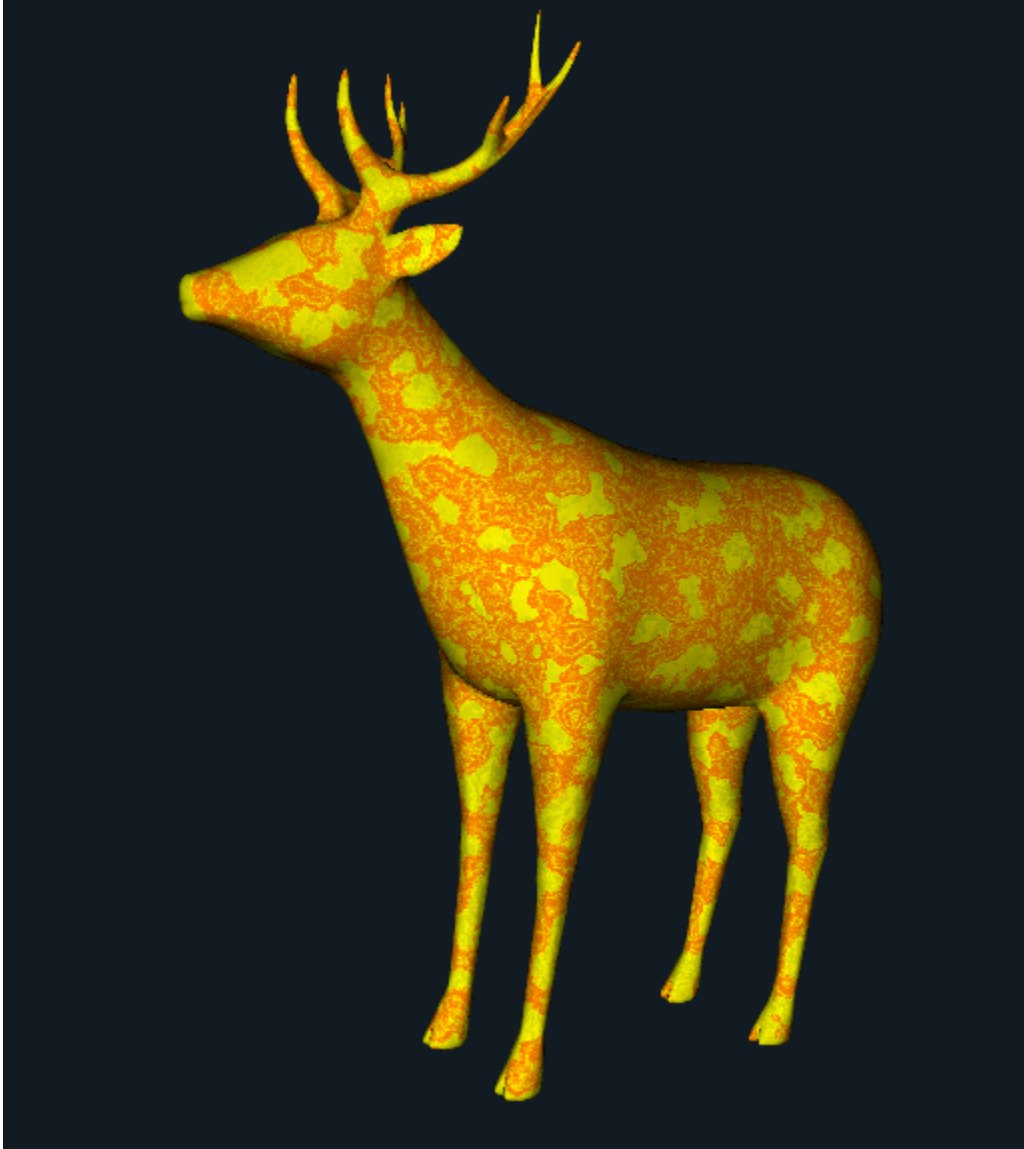
```

{
    float t = (2./3.) * ( z - ChromaRed ) / ( ChromaBlue - ChromaRed );
    t = clamp( t, 0., 2./3. );
    gl_FragColor = vec4( ChromaDepth( t ) , 1. );
}
else
{
    gl_FragColor = mColor;
}
}

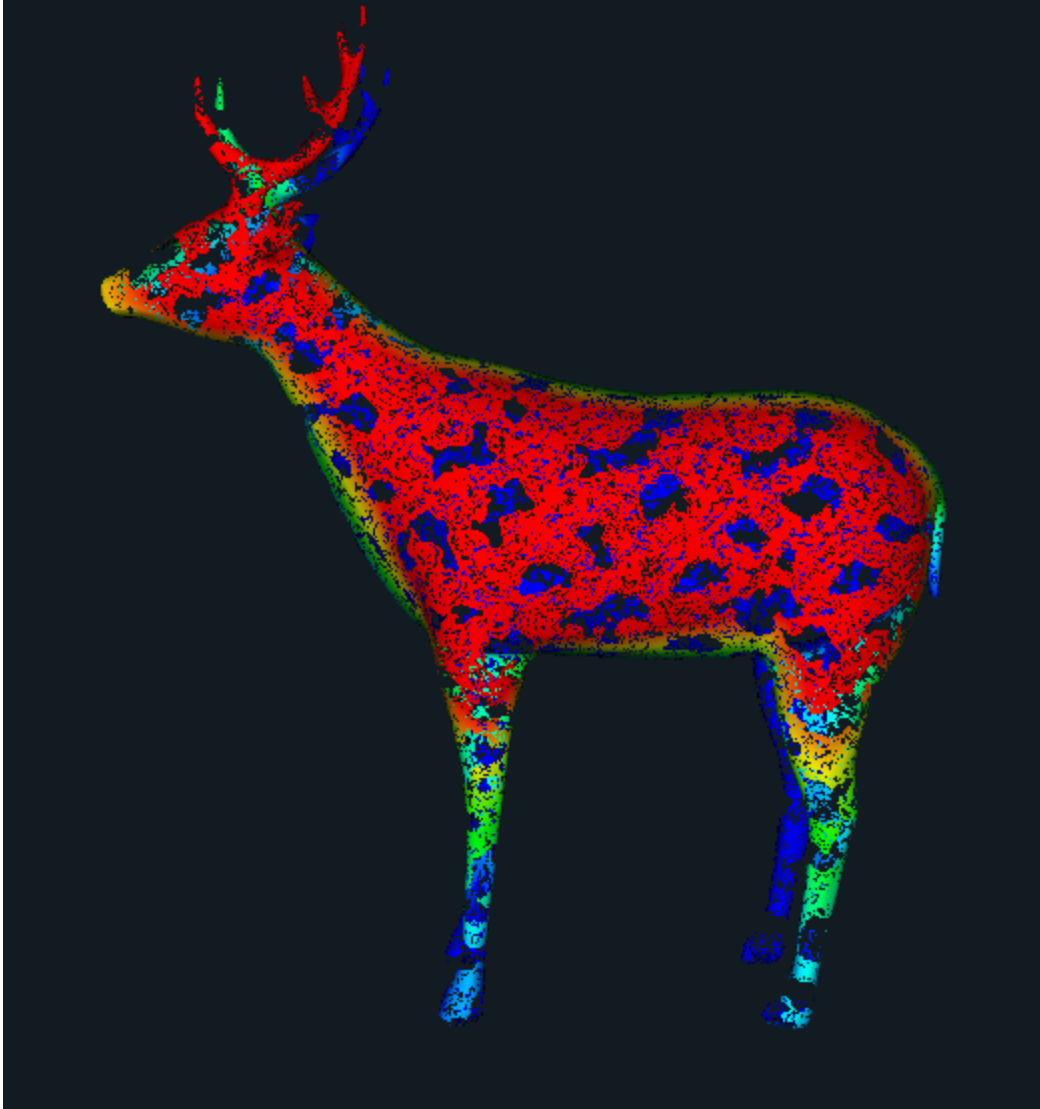
```

the inside loop is for the chromadepth. The reason it is a inside loop is because I only turn the eclipse color to rainbow not include the background color. The range of t is from 0 to 2/3 is because we want the red and blue.

2. Results



Noise and tol implement



The ChromaDepth. I would like to use this one to the poster.