CS 557 -- Winter Quarter 2016

Project #3 Report

Interactive Noisy Elliptical Polka-dots

Name: Kai Shi

ID: 932-504-512

In project3, I used GLSL to write the Noisy Elliptical Polka-dots. Here is what I implemented:

uAd and uBd	True
uNoiseAmp	Ture
uNoiseFreq	True
uTol	True
uAlpha	True
ChromaDepth	True

(1) Code:

Here are my source codes (ovalnoise.glib, ovalnoise.vert, ovalnoise.frag):

ovalnoise.glib:

```
##OpenGL GLIB

LookAt  0 0 3  0 0 0  0 1 0

Perspective 70

MessageBox This implements the Alpha extra credit

MessageBox This implements the ChromaDepth extra credit

Vertex   ovalnoise.vert

Fragment ovalnoise.frag

Program  ovalnoise                                  \

        uAd <.01 .025 .5>
          \

                uBd <.01 .1 .5>                                                 \

        uNoiseAmp <0. 0. 10.>
          \

                uNoiseFreq <0. 1. 10.>                          \
```

```
                uAlpha <0. 1. 1.>                       \
        uTol <0. 0. 1.>                         \
        uUseChromaDepth <false>                     \
        uChromaBlue <-5.  -3.8  0.>                 \
        uChromaRed  <-3.  -2.3  2.>
Color  1  0.5  0.
Sphere
```

ovalnoise.vert:

```
#version 330 compatibility
out vec3  vMCposition;
out vec3  ECpositon;
out float vLightIntensity;
out vec2  vST;
out vec4  vColor;
out float Z_depth;
const vec3 LIGHTPOS = vec3( -2., 0., 10.);
void main( )
{
        vST = gl_MultiTexCoord0.st;
        vec3 tnorm = normalize( vec3( gl_NormalMatrix * gl_Normal ) );
        vec3 ECposition = vec3( gl_ModelViewMatrix * gl_Vertex ).xyz;
        vLightIntensity  = abs( dot( normalize(LIGHTPOS - ECposition), tnorm )  );
        vMCposition = gl_Vertex.xyz;
```

```
	gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
	Z_depth = ECposition.z;
	vColor = gl_Color;
}
```

ovalnoise.frag:

```
#version 330 compatibility
in vec3  vMCposition;
in float  Z_depth;
in float vLightIntensity;
in vec2  vST;
in vec4  vColor;
uniform float uAd;
uniform float uBd;
uniform float uNoiseAmp;
uniform float uNoiseFreq;
uniform float uTol;
uniform float uChromaRed;
uniform float uChromaBlue;
uniform bool uUseChromaDepth;
uniform float uAlpha;
uniform sampler3D Noise3;
uniform bool ChromaDepth;
vec3 Rainbow( 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 );
}
void main()
{
  float s = vST.s;
  float t = vST.t;
  float up = 2*s;
  float vp = t;
  float numinu = floor( up / (2*uAd) );
  float numinv = floor( vp / (2*uBd) );
  float chromaT;
  vec3 TheColor;
  vec4  nv  = texture3D( Noise3, uNoiseFreq * vMCposition );
  float n = nv.r + nv.g + nv.b + nv.a;  // 1. -> 3.
  n = n - 2.;       // -1. -> 1.
  float delta = uNoiseAmp * n;
  float uc = numinu*2*uAd + uAd;
```

```
 float vc = numinv*2*uBd + uBd;
 float d = sqrt(pow((up-uc)/uAd, 2) + pow((vp-vc)/uBd, 2));
 d =d + delta;
float mix_ratio = smoothstep(1-uTol, 1+uTol, d);
vec4 Color = mix(vColor, vec4(1., 1., 1., uAlpha), mix_ratio);
gl_FragColor = Color;
if(uUseChromaDepth)
 {
   float chromaT = (2./3.) * ( Z_depth - uChromaRed ) / ( uChromaBlue - uChromaRed );
   chromaT = clamp( chromaT, 0., 2./3. );
   TheColor = Rainbow( chromaT );
   gl_FragColor.xyz = TheColor;
 }
if (gl_FragColor.a == 0) {
   discard;
}
 gl_FragColor.rgb *= vLightIntensity;
}
```

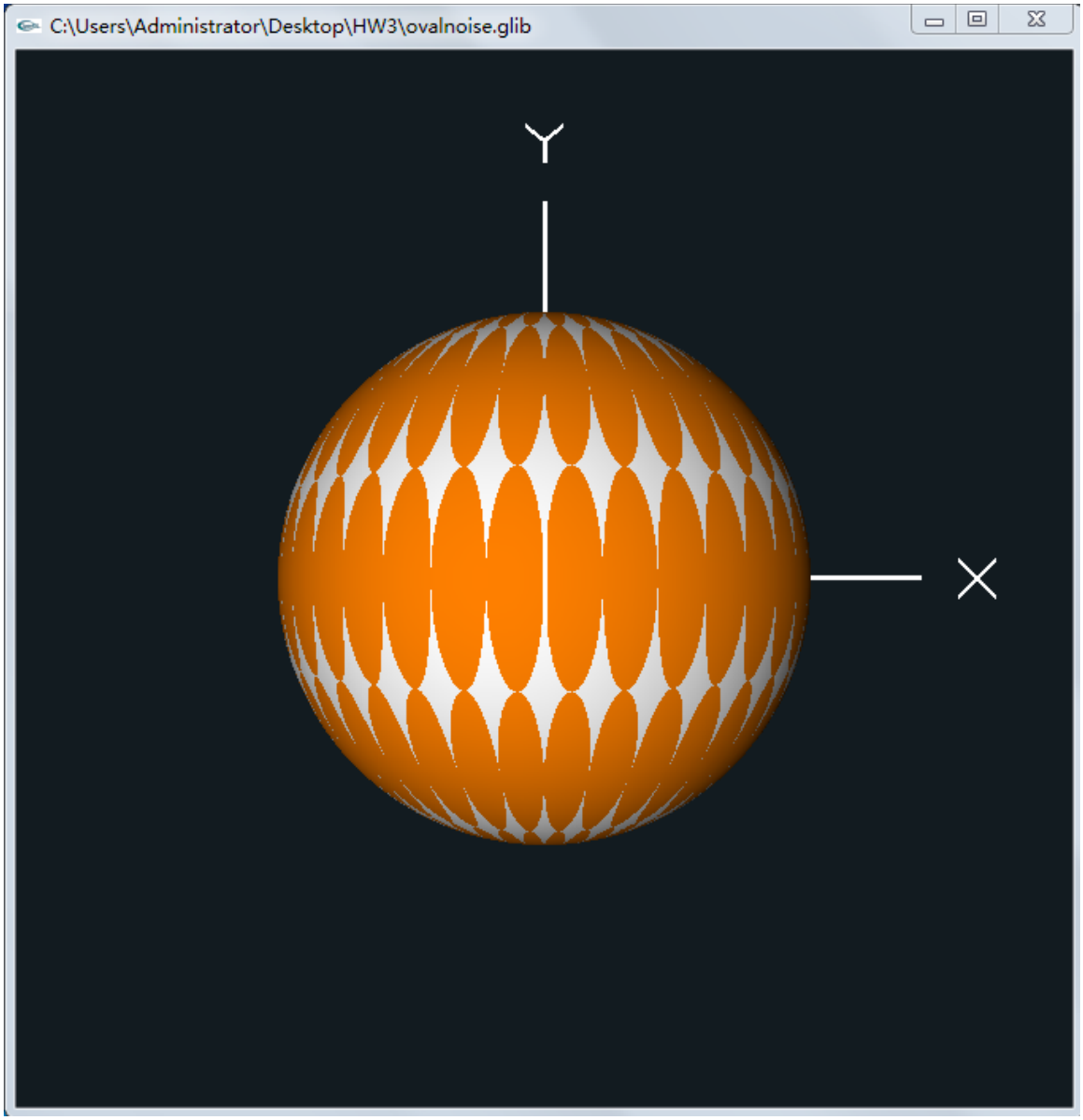
(2) What I did and Reasons

uAd and uBd part: The parameters are passed from glib file and what I need to do is creating numinu and numinv: float numinu = floor(up / (2*uAd)) and float numinv = floor(vp / (2*uBd)). Then I calculate the distance d from center to edge of the ellipse, which is sqrt(pow((up-uc)/uAd, 2) + pow((vp-vc)/uBd, 2)).
For noise part, I use the way from NoiseWithGlman.pdf. uNoiseFreq is a parameter to texture3D function. Then uNoiseAmp just need to multiplies n. Then, add this to the d.
In uTol part, using smoothstep function as required. For color of point located in 1 – uTol and 1+ uTol, we mix orange and white. So, I use mix_ratio = smoothstep(1-uTol, 1+uTol, d) and mix(vColor, vec4(1., 1., 1., uAlpha), mix_ratio) to finish that.

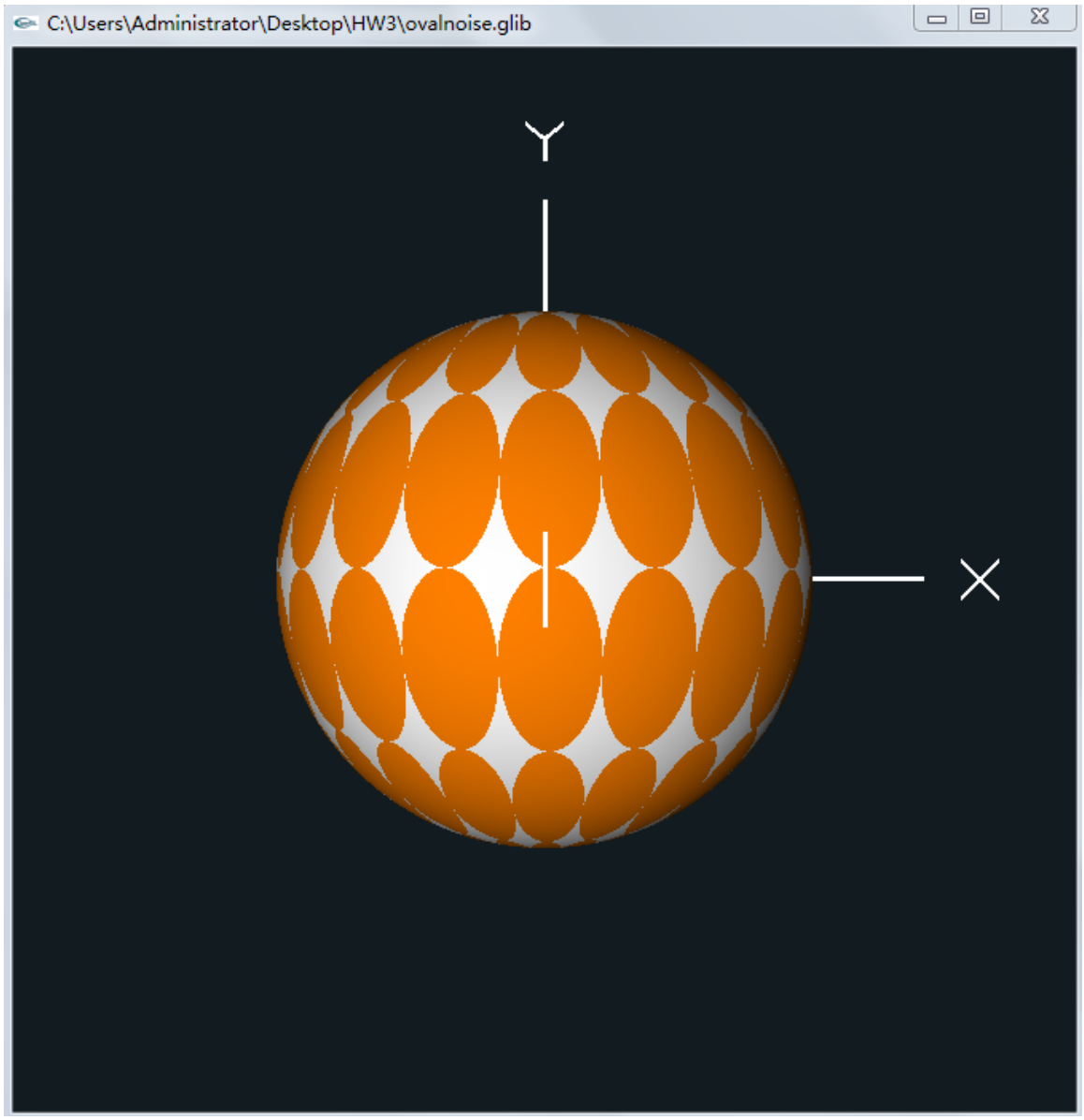
When doing uAlpha part, I put these codes at the end. When gl_FragColor.a ==0, applying discard. This function is only related to alpha value of gl_FragColor.
Last part is ChromaDepth. I used the rainbow function. When choose useChromaDepth, I need to use the ECpostion.z. Because 3D effect, it must in the Eye Coordinates. When we rotate the sphere, the red is still in the front of the sphere. So, I get the ECpostion.z at the vert file and pass it to the frag file. After using the code in the project page, we can get a new color and pass it to the gl_FragColor.xyz (because new color form rainbow function is vec3, so, I can only pass it to gl_FragColor.xyz).

(3) Results

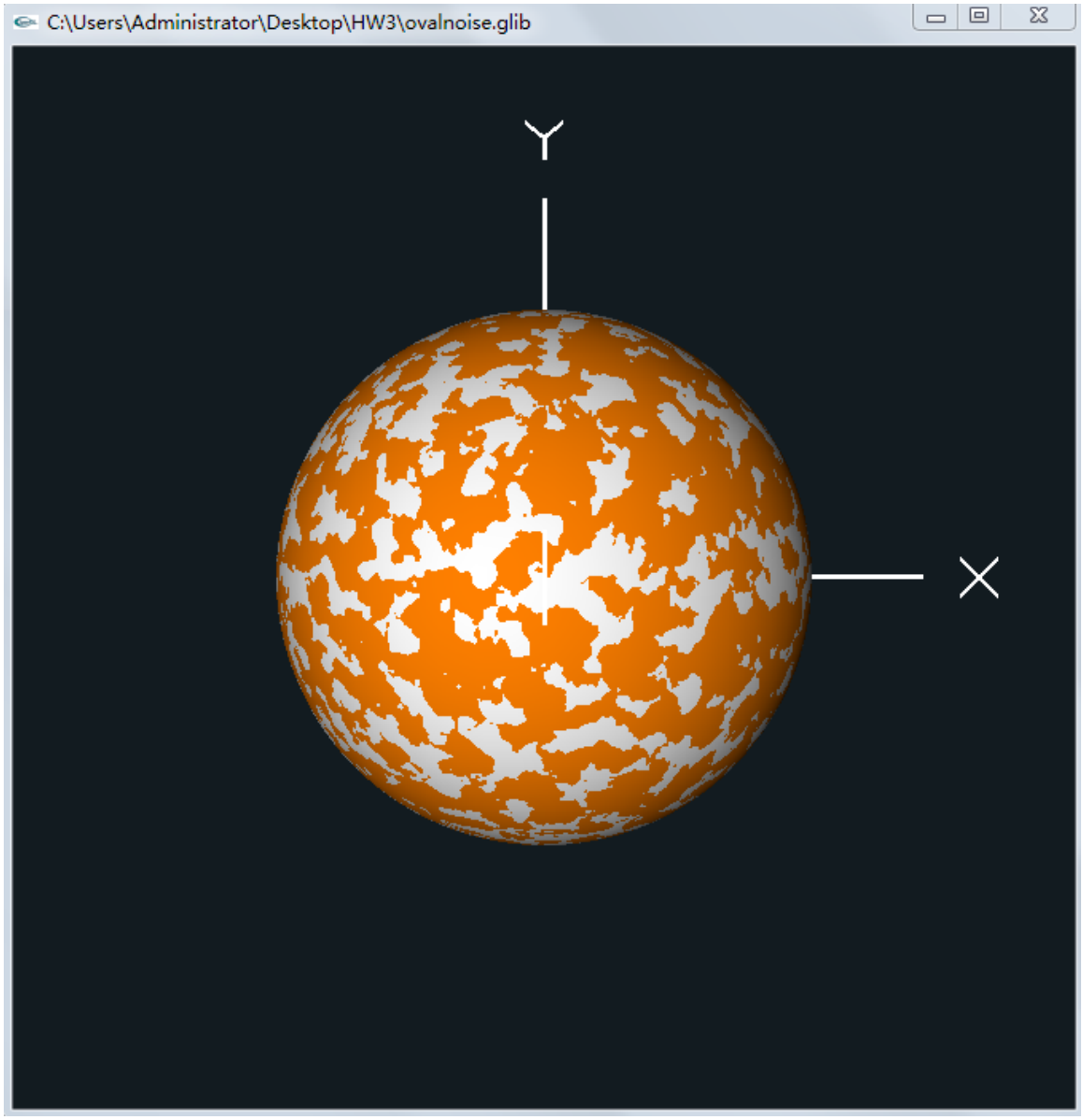
Only ellipses:



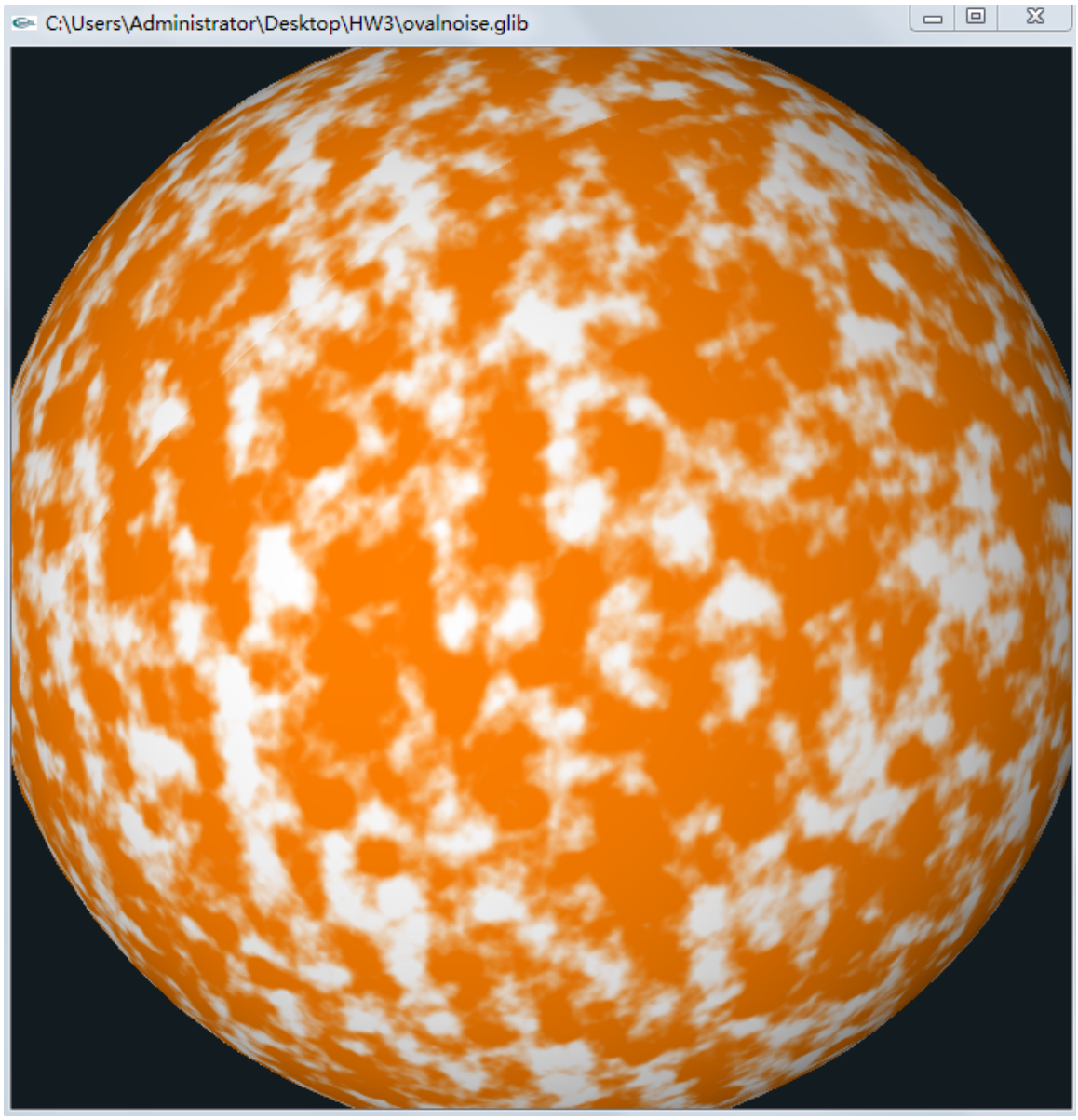
Changing uAd and uBd:



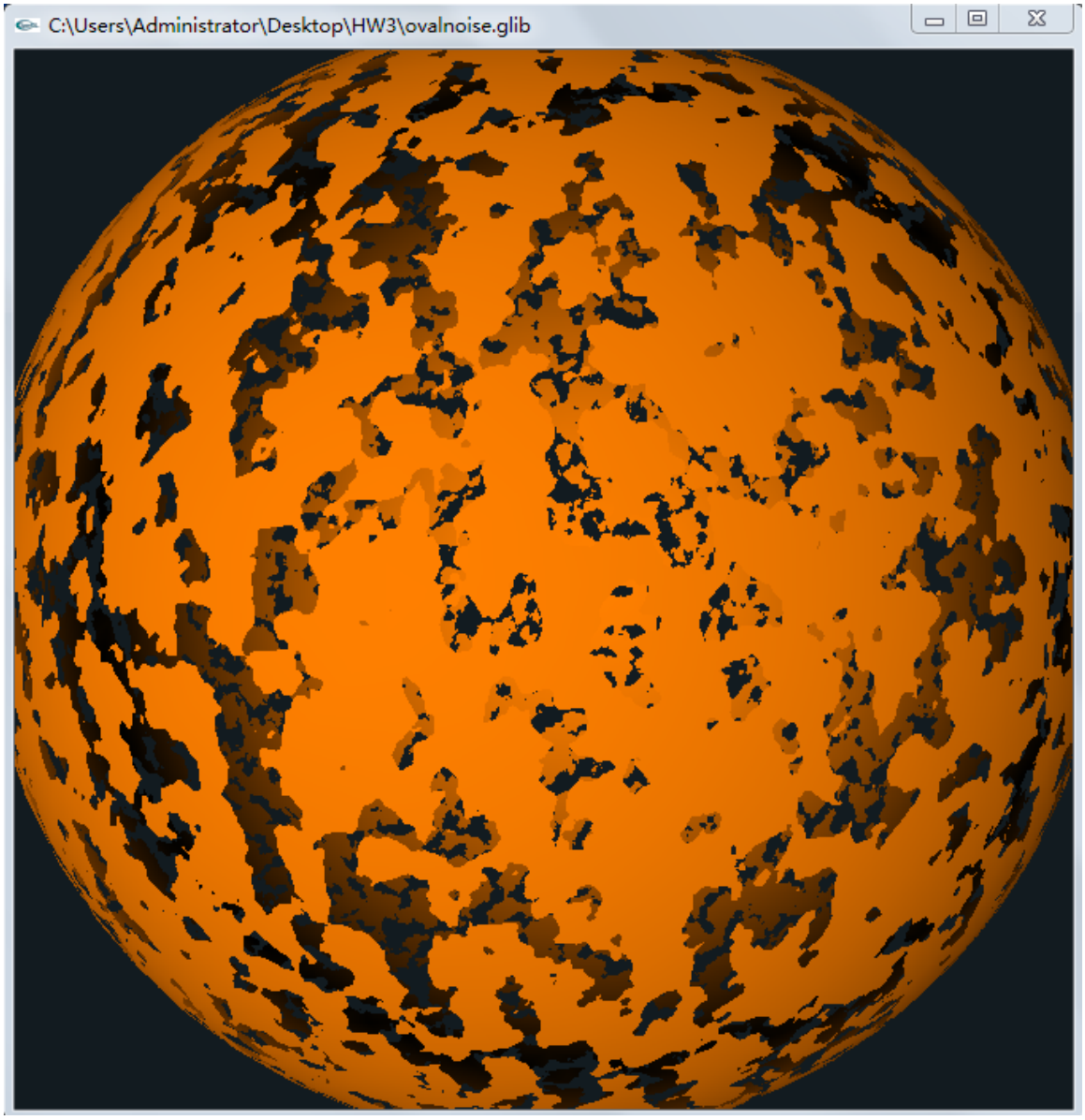
Noise:



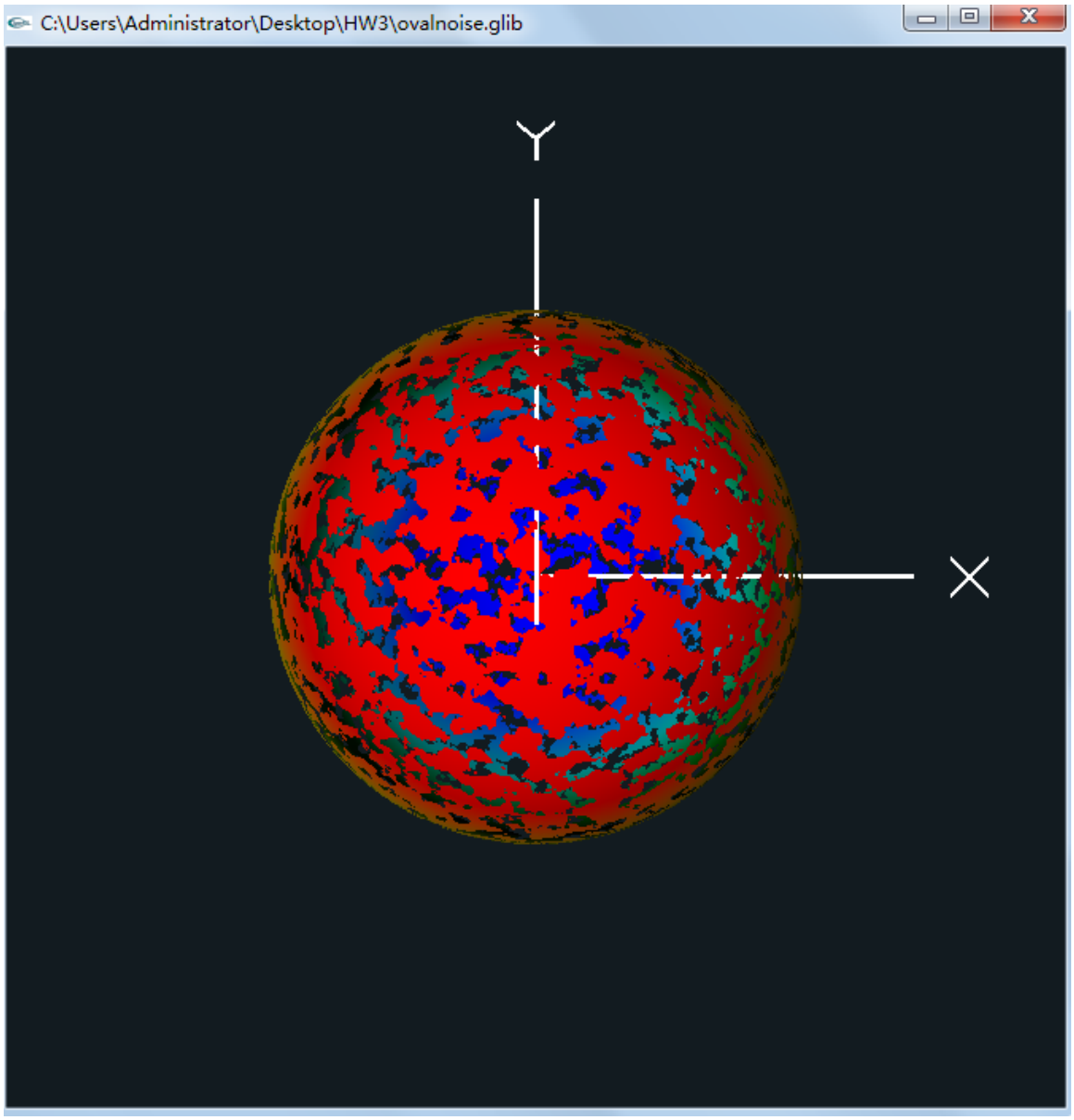
uTol:



uAlpha:



ChormaDepth:



When rotating, the red is still in the front and blue is in the back: