CS 557 -- Winter Quarter 2016

Project #2 Report

Noisy Displaced Elliptical Dots

Name: Kai Shi

ID: 932-504-512

In project2, I turn Project #1's elliptical dots into more random-looking height "islands".

(1) Code:

Here are my source codes (ellipses.rib, ellipsesnoise.sl, ellipsesnoisedisp.sl):

ellipses.rib:

```
##RenderMan RIB

version 3.03

# declare the variables:

Declare "Ad" "uniform float"

Declare "Bd" "uniform float"

# define the output file:

Display "ovals.tiff" "file" "rgb"

Format 512 512 -1

ShadingRate 1

# define the lighting:

LightSource "ambientlight" 1 "intensity" [0.25]

LightSource "distantlight" 2 "intensity" [0.75] "from" [5 8 -10] "to" [0 0 0]

# define the rendering parameters:

Projection "perspective" "fov" [70]

# define the scene to be rendered:

WorldBegin

        Translate 0 0 6

        Attribute "bound" "displacement" [1.5]

        Surface "ovals" "Ad" 0.025 "Bd" 0.10 "height" 0.10
        Displacement "ovalnoised" "Ad" 0.025 "Bd" 0.10 "height" 0.10


        Color   [1 1 1]                         # specify the Cs color

        Opacity [1 1 1]                                 # specify the Os opacity

        TransformBegin

                Rotate 90  1. 0. 0.             # rotate so don't see north pole

                Sphere 3 -3 3 360               # a full sphere

        TransformEnd

WorldEnd
```

ellipsesnoise.sl:

```
surface
ovals(
        float   Ad  = 0.025,
                Bd = 0.10,                              // probability of seeing orange
                Ks = 0.5,
                Kd = 0.5,                               // diffuse  coefficient
                Ka = 0.1,                               // ambient  coefficient
                roughness = 0.1;                        // specular roughness
        color   specularColor = color( 1, 1, 1 )        // specular color
)
{
        varying vector Nf = faceforward( normalize( N ), I );
        vector V = normalize( -I );

        float up = 2. * u;      // because we are rendering a sphere
        float vp = v;
        float numinu = floor( up / (2*Ad) );
        float numinv = floor( vp / (2*Bd) );

        color dotColor = Cs;

                                // noise magnitude
                                point PP = point "shader" P;
                                float magnitude = 0.;
                                float size = 1;   //like noiseFreq
                                float i;
                                for( i = 0.; i < 6.0; i += 1.0 )
                                {
                                    magnitude += 3*(noise(3* size * PP ) - 0.5 ) / size;
                                    size *= 2.0;
                                }

                                float uc = numinu*2*Ad + Ad;
                                float vc = numinv*2*Bd + Bd;
                                up = (up - uc)/Ad;
                                vp = (vp - vc)/Bd;
                                point upvp = point( up, vp, 0. );
                                point cntr = point( 0., 0., 0. );

                                vector delta = upvp - cntr;

                                float oldrad = length(delta);
                                float newrad = oldrad + magnitude;
                                delta = delta * newrad / oldrad;

                                float deltau = xcomp(delta) ;
                                float deltav = ycomp(delta);

                                up = deltau;
                                vp = deltav ;


                                float ellipseEquation= up*up +vp *vp;

                                if( ellipseEquation <= 1. )
                                {
                                        dotColor = color( 1., .5, 0. );
                                }


        Oi = 1.;
        Ci = Oi * ( dotColor * ( Ka * ambient() + Kd * diffuse(Nf) ) + specularColor * Ks
* specular( Nf, V, roughness ));
}
```

ellipsesnoisedisp.sl:

```
displacement
ovalnoised(
    float    Ad  = 0.025,
            Bd = 0.10,                              // probability of seeing orange
            Ks = 0.5,
            Kd = 0.5,                               // diffuse  coefficient
            Ka = 0.1,                               // ambient  coefficient
            height = 0.1 ,
            DispAmp = 0.20,         // displacement amplitude
            roughness = 0.1;                        // specular roughness
    color   specularColor = color( 1, 1, 1 )        // specular color
)
{

    float disp = 0.0 ;

    float up = 2. * u;      // because we are rendering a sphere
    float vp = v;
    float numinu = floor( up / (2*Ad) );
    float numinv = floor( vp / (2*Bd) );

                            // noise magnitude
                            point PP = point "shader" P;
                            float magnitude = 0.;
                            float size = 1.;
                            float i;
                            for( i = 0.; i < 6.0; i += 1.0 )
                            {
                                    magnitude += 3* ( noise( 3* size * PP ) - 0.5 ) /size;
                                    size *= 2.0;
                            }

                            float uc = numinu*2*Ad + Ad;
                            float vc = numinv*2*Bd + Bd;
                            up = (up - uc)/Ad;
                            vp = (vp - vc)/Bd;
                            point upvp = point( up, vp, 0. );
                            point cntr = point( 0., 0., 0. );

                            vector delta = upvp - cntr;

                            float oldrad = length(delta);
                            float newrad = oldrad +  magnitude;
                            delta = delta * newrad / oldrad;

                            float deltau = xcomp(delta) ;
                            float deltav = ycomp(delta);

                            up = deltau;
```

```
            vp = deltav;
            float ellipseEquation = up*up+vp*vp;

            if( ellipseEquation <=1. )
            {
            disp = height- height* ellipseEquation;
        // float t = smoothstep( 0., Amp, disp);
    //disp = t*disp ;                            // apply the blending

            }

if( disp != 0. )
{

    P= P + normalize(N) * disp;
    N = calculatenormal(P);
    //normal n = normalize(N);
    //N = calculatenormal( P + disp * n );
}

varying vector Nf = faceforward( normalize( N ), I );
vector V = normalize( -I );
```

(2) What I did and Reasons

The first thing a need to is adding noise in the project#1. I set noise magnitude += NoiseMag* (noise(NoiseFreq* size * PP) - 0.5) / size. Also, I set a vector delta, which from cntr to the current point upvp. So, the length of the delta is called oldrad, then, I add noise magnitude to the old rad. The vector delta now is delta * newrad / oldrad. Using x, y of the delta in the ellipse equation. If the distance <=1, let the color be beaver orange. Result is noise only picture.

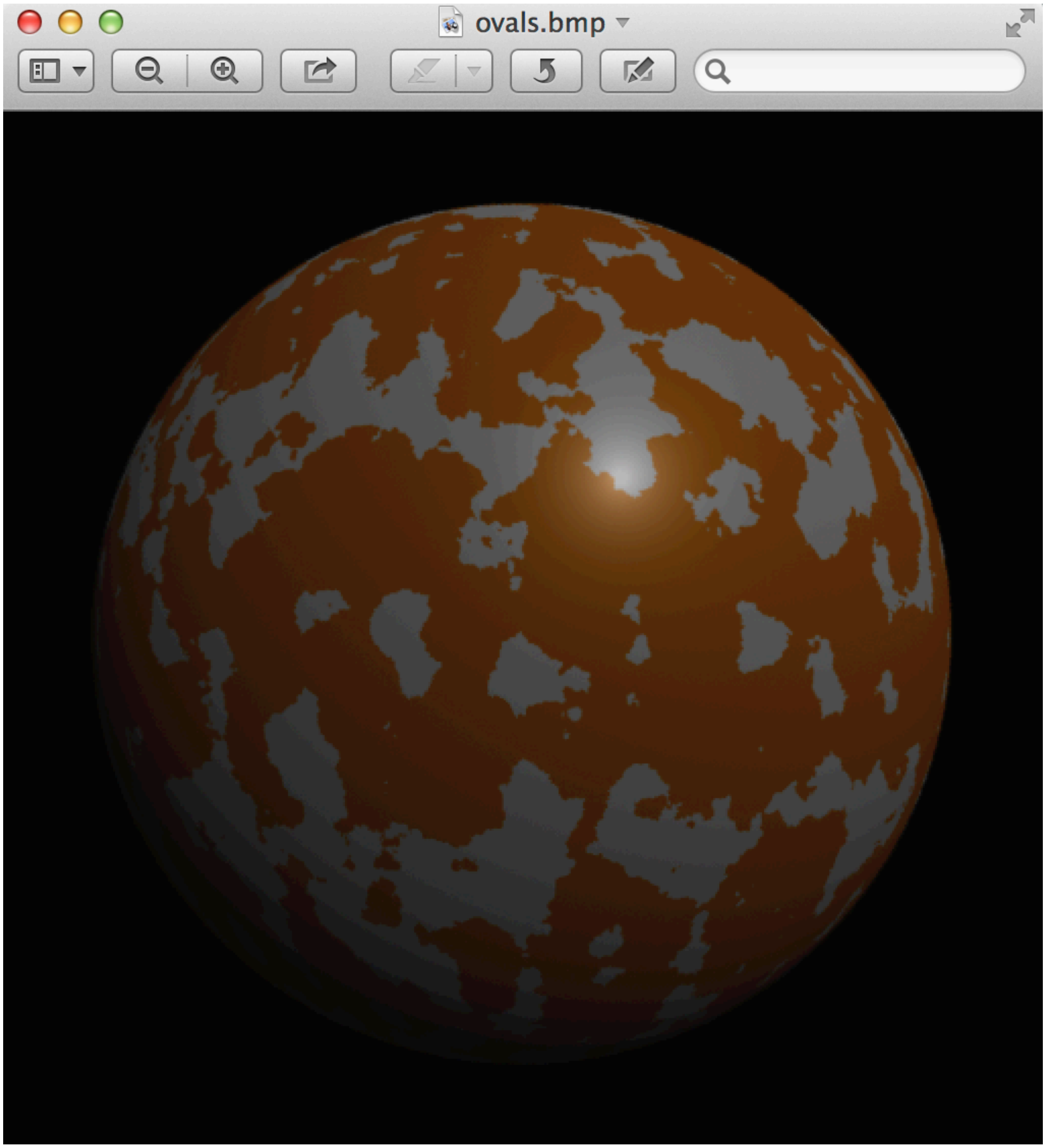
Next, after finishing surface render, I write the displacement shader. In this shader, I set a disp, and let disp equals to (1- ellipseEquation) * height, (ellipseEquation is d we learned in class). Then, using P= P + normalize(N) * disp; N = calculatenormal(P) to apply displacement mapping. At the beginning, color can not cover the whole islands. Then, I use the smoothstep to make the edge more smooth, and the result becomes better, but I don’t know why. According to professor, it doesn’t need to use smoothstep, so, I delete it again.

Finally, I use N = calculatenormal(P + disp * n) to apply bump mapping. Islands in the edge of the sphere lower, but the color matches island better. However, it seems not very realistic.

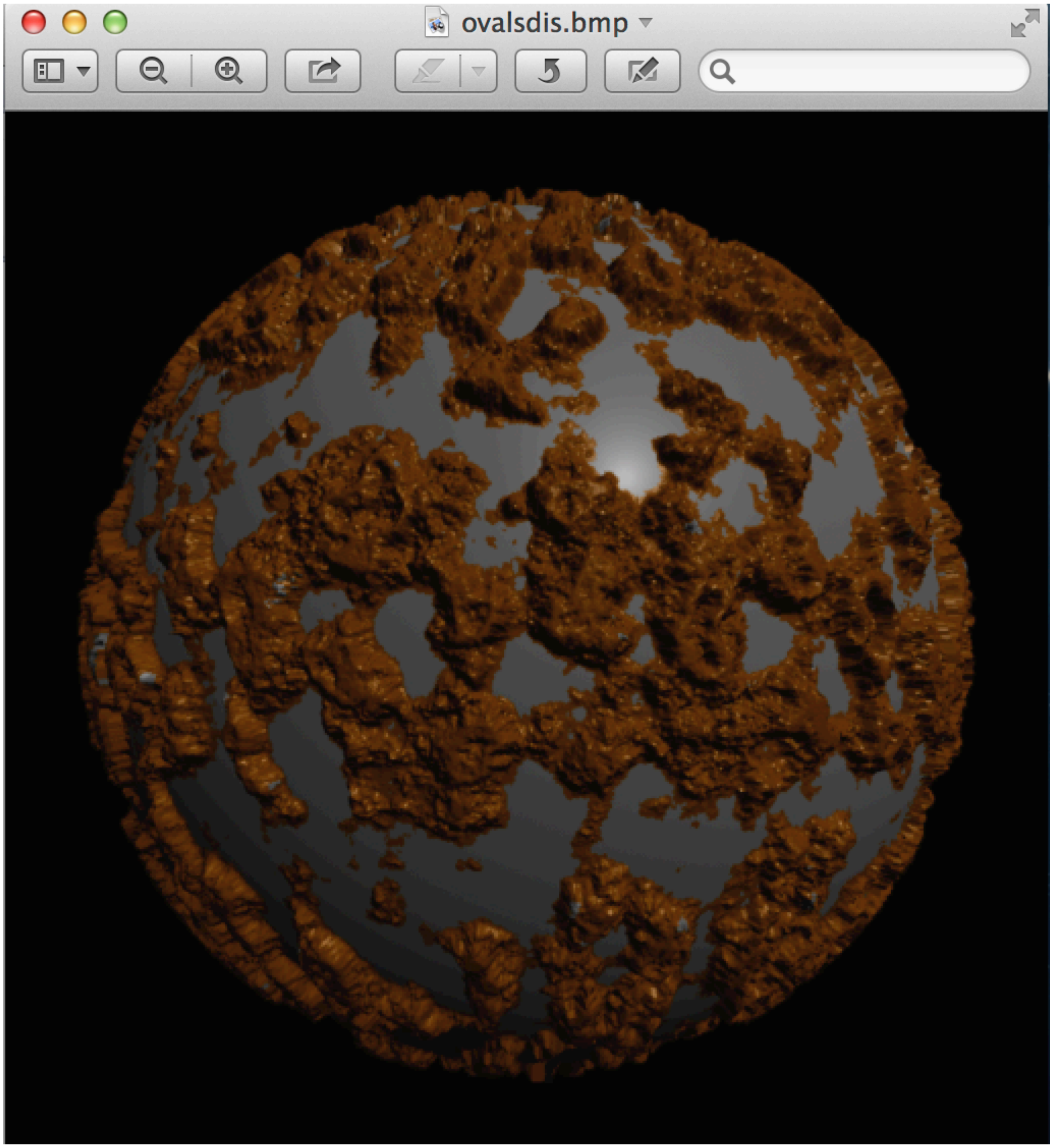
(3) Results

The results are like this:

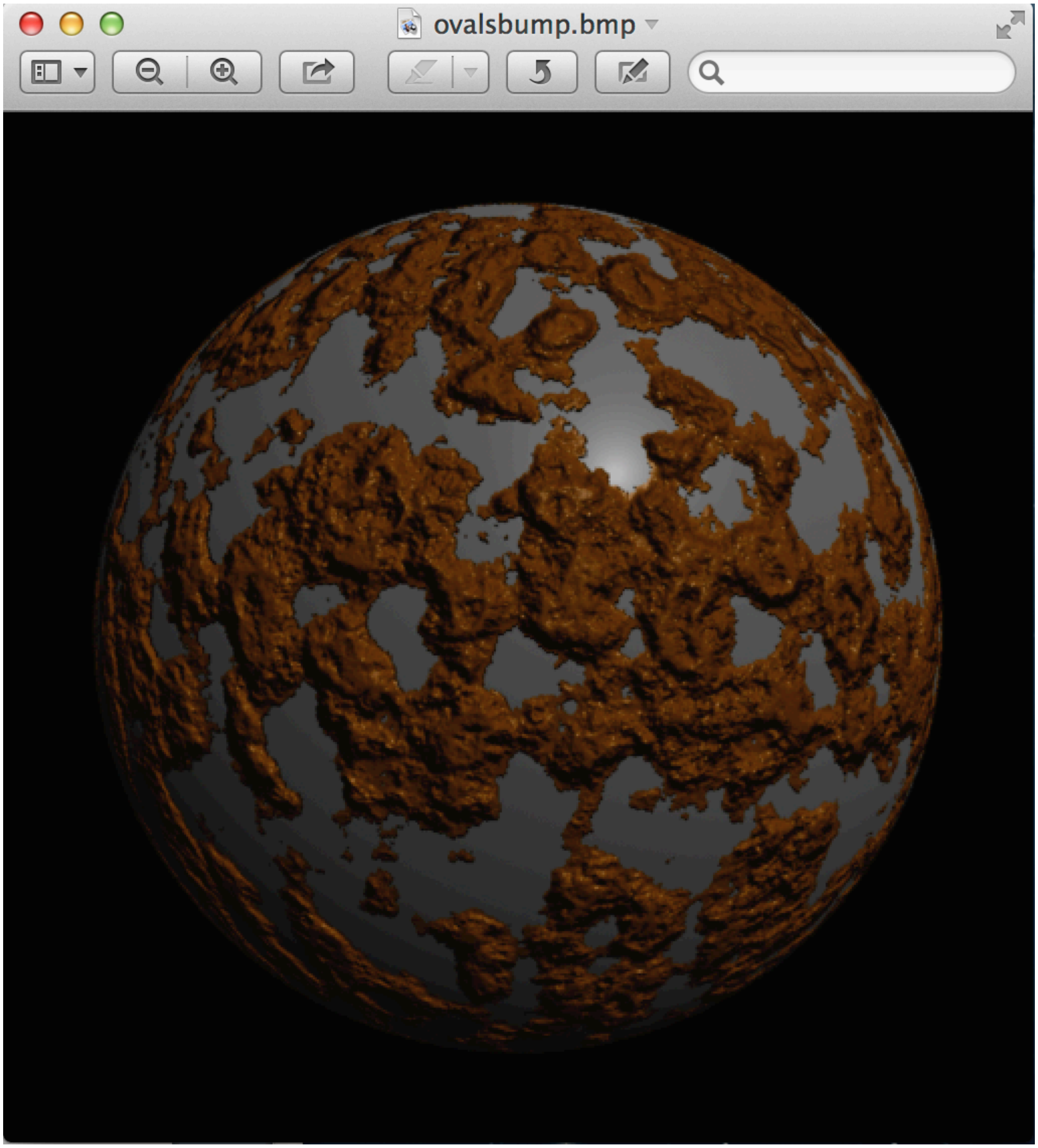
This is noise only result:



This is the displacement mapped result:



This is the bump mapped result:


ovalsbump.bmp