Natasha Anisimova

Project # 2

Noisy Displaced Elliptical Dots

**Rib File**

##RenderMan RIB

version 3.03

Declare "Height" "uniform float"

Declare "Ramp" "uniform float"

Declare "Width" "uniform float"

Display "second.tiff" "file" "rgb"

Format 500 500 -1

LightSource "ambientlight" 1 "intensity" [0.25]

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

ShadingRate 1

Projection "perspective" "fov" [70]

WorldBegin

Attribute "bound" "displacement" [1.5]

Surface "seconds" "Width" 0.10

# specify the surface shader

Displacement "secondd" "Width" 0.10 "Height" 0.2 "Ramp" 0.20

# specify the displacement shader

Color [1 1 1] # Cs

Opacity [1 1 1] # Os

TransformBegin

Translate 0 0 6 # move away from the eye

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

Sphere 3 -3 3 360 # a full sphere

TransformEnd

WorldEnd

**Second.sl**

displacement

secondd(

float Width = 0.10, // square width

Ramp = 0.1, // fraction of square used in the ramp

Height = 0.2, // displacement height

Ad = 0.50,

Bd = 0.35,

NoiseAmp = 0.00,

DispAmp = 0.10;

)

{

float TheHeight = 0.; // how much displacement to apply

float up = 2. \* u;

float vp = v;

float numinu = floor( up / Ad ); //Ad

float numinv = floor( vp / Bd ); //Bd

point PP = point "shader" P;

float magnitude = 0.;

float size = 1.;

float i;

for( i = 0.; i < 6.0; i += 1.0 )

{

magnitude += ( noise( size \* PP ) - 0.5 ) / size;

size \*= 2.0;

}

float uc = Ad \*numinu + (Ad/2);

float vc = Bd \*numinv+ (Bd/2);

float r = Width/2.;

float Ar = Ad/2.;

float Br = Bd/2.;

float du = up - uc;

float dv = vp - vc;

float oldrad = sqrt( du\*du + dv\*dv );

float newrad = magnitude + oldrad;

float factor = newrad/oldrad;

du \*= factor;

dv \*= factor;

float d = pow((du)/Ar, 2.) + pow((dv)/Br, 2.);//pow((up - uc)/Ar, 2.) + pow((vp - vc)/Br, 2.);

if ( d <= 1.)

{

float umin = numinu\*Ad; // square boundaries in u

float umax = umin+Ar;

float vmin = numinv\*Bd; // square boundaries in v

float vmax = vmin+Br;

float distu = min( up-Ar, umax-Ar ); // dist to nearest u boundary

float distv = min( vp-Br, vmax-Br ); // dist to nearest v boundary

float dist = min( distu, distv )/r; // dist to nearest boundary

float t = 1. - d; //smoothstep( 0., Ramp, dist ); // 0. if dist <= 0., 1. if dist >= Ramp

//float t = smoothstep( 0., Ramp, dist);

TheHeight = t\*Height; // apply the blending

}

#define DISPLACEMENT\_MAPPING

float disp = 1. - d;

if( disp != 0. ) //disp

{

#ifdef DISPLACEMENT\_MAPPING

P = P + normalize(N) \* TheHeight;

N = calculatenormal(P);

#else

N = calculatenormal( P + normalize(N) \* TheHeight );

#endif

}

}

**Seconds.sl**

surface

seconds(

float Width = 0.10, // square width

Ks = 0.4, // specular coefficient

Kd = 0.5, // diffuse coefficient

Ka = 0.1, // ambient coefficient

Ad = 0.50,

Bd = 0.35,

Roughness = 0.1; // specular roughness

color SpecularColor = color( 1, 1, 1 ) // specular color

)

{

color ORANGE = color( .1, .7, .1 );

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

vector V = normalize( -I );

float up = 2. \* u;

float vp = v;

float numinu = floor( up / Ad );

float numinv = floor( vp / Bd );

Oi = Os; // use whatever opacity the rib file gave us

color TheColor = Cs;

//float d =

//if( mod( numinu+numinv, 2. ) == 0 )

point PP = point "shader" P;

float magnitude = 0.;

float size = 1.;

float i;

for( i = 0.; i < 6.0; i += 1.0 )

{

magnitude += ( noise( size \* PP ) - 0.5 ) / size;

size \*= 2.0;

}

float uc = Ad \*numinu + (Ad/2);

float vc = Bd \*numinv+ (Bd/2);

float r = Width/2;

float Ar = Ad/2.;

float Br = Bd/2.;

float du = up - uc;

float dv = vp - vc;

float oldrad = sqrt( du\*du + dv\*dv );

float newrad = magnitude + oldrad;

float factor = newrad/oldrad;

du \*= factor;

dv \*= factor;

if ( pow((du)/Ar, 2.) + pow((dv)/Br, 2.)<= 1.)

TheColor = ORANGE;

else

Oi = color( 0.6, 0.6, 0.6 );

Ci = TheColor \* Ka \* ambient();

Ci = Ci + TheColor \* Kd \* diffuse(Nf);

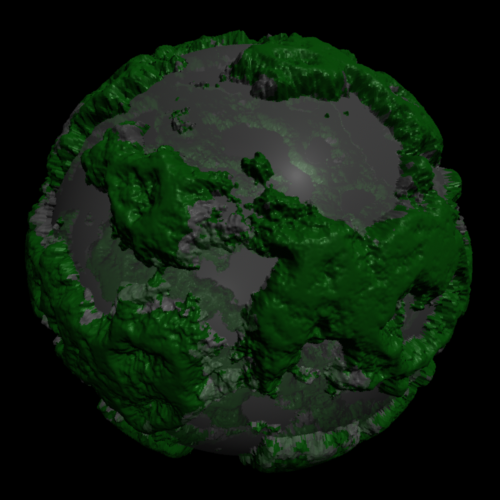
Ci = Ci + SpecularColor \* Ks \* specular( Nf, V, Roughness );

Ci = Ci \* Oi;

}

**Explanation**

Essentially what happens is that I create ellipses and then add noise. The seconds file is just adding noise to the surface. The second file adds on the displacement. I changed the displacement if statement to do bump-mapping. I could also add noise in the negative direction to make it look like the ocean of my world also has different heights but I did not like the way it looked. There was too much noise. I tried to fix the little gray parts that were displaced but I wasn’t entirely sure what was causing it to begin with.



C:\Users\Cepribro\Downloads\second (1).tiff

C:\Users\Cepribro\Downloads\second (2).tiff

For bump-mapping

#define DISPLACEMENT\_MAPPING

float disp = 1. - d;

if( disp != 0. && disp > 0 ) //disp

{

#ifdef DISPLACEMENT\_MAPPING

//P = P + normalize(N) \* TheHeight;

//N = calculatenormal(P);

normal n = normalize(N);

N = calculatenormal( P + disp \* n );

#else

N = calculatenormal( P + normalize(N) \* TheHeight );

#endif

}

}