CS557 Project #2

Yunfan Li

[liyunf@oregonstate.edu](mailto:liyunf@oregonstate.edu)

1. **Source Listings**
   1. ***a2.rib***

##RenderMan RIB

version 3.03

Declare "Ar" "uniform float"

Declare "Br" "uniform float"

Display "a2.tiff" "file" "rgb"

Format 512 512 -1

ShadingRate 1

LightSource "ambientlight" 1 "intensity" [0.25]

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

Projection "perspective" "fov" [70]

WorldBegin

Translate 0 0 6

Attribute "bound" "displacement" [1.]

Surface "a2s" "Ar" 0.065 "Br" 0.175

Displacement "a2d" "Ar" 0.065 "Br" 0.175

Color [1 1 1]

Opacity [1 1 1]

TransformBegin

Rotate 90 1. 0. 0.

Sphere 3 -3 3 360

TransformEnd

WorldEnd

* 1. ***a2s.sl***

surface

a2s (float

Ar=0.4,

Br=0.4,

Ks=0.5,

Kd=0.5,

Ka=0.1,

roughness=0.1;

color specularcolor=color(1, 1, 1))

{

float up = 2. \* u;

float vp = v;

float numinu = floor( up / (2 \* Ar) );

float numinv = floor( vp / (2 \* Br) );

float uc = numinu \* 2. \* Ar + Ar;

float vc = numinv \* 2. \* Br + Br;

point PP = point "shader" P;

float magn = 0.2;

float size = 1.05;

float i;

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

{

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

size \*= 2.0;

}

point uvp = point(up, vp, 0.);

point uvc = point(uc, vc, 0.);

vector delta = uvp-uvc;

float oldrad = pow((up-uc)/Ar, 2) + pow((vp-vc)/Br, 2);

float newrad = oldrad + magn;

delta = delta \* newrad/oldrad;

float deltau = xcomp(delta);

float deltav = ycomp(delta);

float d = pow(deltau/Ar, 2) + pow(deltav/Br, 2);

color dotColor = Cs;

if (d <= 1. ) {

dotColor = (1., .5, 0.);

}

else {

dotColor = (0., 0.5, 1.);

}

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

vector V = normalize( -I );

Oi = 1.;

Ci = Oi \* ( dotColor \* ( Ka \* ambient() + Kd \* diffuse(Nf) ) + specularcolor \* Ks \* specular( Nf, V, roughness ) );

}

* 1. ***a2d.sl***

displacement

a2d (float

Ar=0.1,

Br=0.1)

{

float disp = 0.;

float up = 2. \* u;

float vp = v;

float numinu = floor( up / (2. \* Ar) );

float numinv = floor( vp / (2. \* Br) );

float uc = numinu \* 2. \* Ar + Ar;

float vc = numinv \* 2. \* Br + Br;

point PP = point "shader" P;

float magn = 0.2;

float size = 1.05;

float i;

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

{

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

size \*= 2.0;

}

point uvp = point(up, vp, 0.);

point uvc = point(uc, vc, 0.);

vector delta = uvp-uvc;

float oldrad = pow((up-uc)/Ar, 2) + pow((vp-vc)/Br, 2);

float newrad = oldrad + magn;

delta = delta \* newrad/oldrad;

float deltau = xcomp(delta);

float deltav = ycomp(delta);

float d = pow(deltau/Ar, 2) + pow(deltav/Br, 2);

if (d <= 1.) {

disp = (1. - d) \* 0.1;

if (disp != 0.)

{

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

N = calculatenormal(P);

}

}

}

1. **What I did & Why it worked**

***2.1 What I did***

In my a2.rib file, I created two “uniform float” variables named “Ar”and “Br” respectively. Then tiff file was denoted as my output file format and the color mode of output file is RGB, the width and height of output file were both 512 with square pixels. Next two lines was used to create a specific light source so that any objects in this scene became more real. In next line, the projection of this scene was declared as “perspective”. After “WorldBegin”, I translate the canvas to (0, 0, 6), then extend bounding box by 1.0. “a2s.sl” and “a2d.sl” were specified as my suface shader and displacement shader after the bounding box was extended. At the same time, two variables whose name is “Ar” and “Br” were assigned their value and passed to surface shader and displacement shader respectively. Then the color and opacity of objects in this scene were set up. From “TransformBegin” to “TransformEnd”, a sphere of this scene was created and rotated 90 degrees on X-axis.

In my a2s.sl file, global variable “u” and “v” were firstly used to calculate which box does this point belong to, “u” was timed 2.0 because it just covers 180 degrees instead of 360 degrees which “v” covers. Then the center point coordinate of this box was calculated using “numinu” and “numinv”. Then a noise variable “magn” was calculated through for-loop. And normal radius value of this point was generated by using ellipse equation. The new noised radius was calculated by adding normal radius and noise variable together. And the vector which point from center point to current point would multiply a rate which came from new radius over normal radius. Then a new “u” and a new “v” was extracted from new “delta” vector. This new “u” and new “v” was used in ellipse equation again to calculate a fake radius that can act as a noise.

In my a2d.sl file, the code is basically as same as a2s.sl. This is because I need displacement shader and surface shader act as work together. So the noise part is exactly same but the displacement was calculated based on the new radius.

***2.2 Why it worked***

The noise is a variable that we can control. In my surface shader and displacement shader, there is exactly same code existing in these two files to calculate noise variable and generate newly fake point coordinates pair. This step ensures that the part of what we painted with different color is the same as part of what we displaced. So after shading with these two shaders, our sphere looks like a water planet with a few of barren island standing on the surface of ocean.

1. **Images**



Figure Surface Noise Only

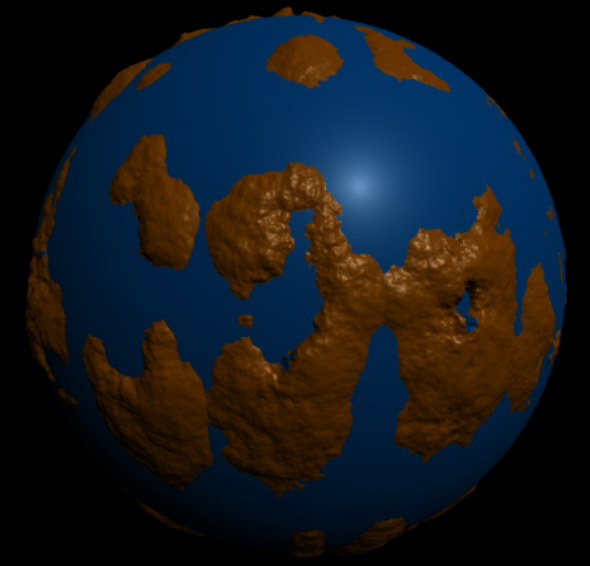


Figure Surface and Displacement Noise (Displacement-Mapped)

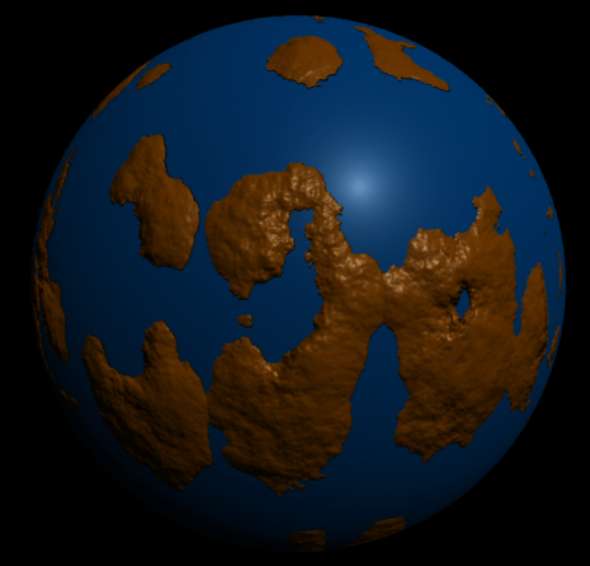


Figure Surface and Displacement Noise (Bump-Mapped)