Project 2

Alan Neads

30th January, 2017

noisy_sphere.rib

```
##RenderMan RIB
version 3.03

Declare "Ad"    "uniform float"
Declare "Bd"    "uniform float"
Declare "NoiseAmp"   "uniform float"
Declare "DispAmp"     "uniform float"

Display "noisy_sphere.tiff" "file" "rgb"
Format 1024 1024 -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
        Surface "noisy_sphere" "Ad" 0.100 "Bd" 0.050
        Color [0.2 0.2 1]
        Opacity [1 1 1]
        TransformBegin
                Rotate 90 1. 0. 0.
                Sphere 3 -3 3 360
        TransformEnd
WorldEnd
```

noisy_sphere.sl

```
surface
noisy_sphere( float Ad = 0.100, Bd = 0.050, Ks = 0.5, Kd = 0.5, Ka = .1, roughness = 0.1; color
specularColor = color( 1, 1, 1 ) ) {
	point PP = point "shader" P;
	float magnitude = 0.;
	float size = 1.0;
	float i;
	for( i = 0.; i < 6.0; i += 1.0 )
	{
		magnitude += (noise(size * PP) - 0.5) / size;
		size *= 2.0;
	}

	float Ar = Ad / 2.0;
	float Br = Bd / 2.0;

	float up = 2. * u;
	float vp = v;

	float numinu = floor( up / Ad );
	float numinv = floor( vp / Bd );

	float uc = numinu * Ad + Ar;
	float vc = numinv * Bd + Br;

	float du = up - uc;
	float dv = vp - vc;

	float factor = 4.0 * magnitude;

	float d = (((0.0 - du) / Ar) * ((0.0 - du) / Ar)) + (((0.0 - dv) / Br) * ((0.0 - dv) / Br));

	color dotColor = Cs;
	if( d + factor <= 1.0 ) {
		dotColor = color( 1.0, 0.5, 0. );
	}

	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));
}
```

noisy_displaced_sphere.rib

```
##RenderMan RIB
version 3.03

Declare "Ad"    "uniform float"
Declare "Bd"    "uniform float"
Declare "NoiseAmp"   "uniform float"
Declare "DispAmp"     "uniform float"

Display "noisy_displaced_sphere.tiff" "file" "rgb"
Format 1024 1024 -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
        Displacement "noisy_displaced_sphere" "Ad" 0.100 "Bd" 0.050
        Surface "noisy_sphere" "Ad" 0.100 "Bd" 0.050
        Color [0.2 0.2 1]
        Opacity [1 1 1]
        TransformBegin
                Rotate 90 1. 0. 0.
                Attribute "displacementbound" "sphere" [3] "coordinatesystem" ["world"]
                Sphere 3 -3 3 360
        TransformEnd
WorldEnd
```

noisy_displaced_sphere.sl

```
displacement
noisy_displaced_sphere( float
            Ad = 0.100,             // u diameter
            Bd = 0.050,              // v diameter
            DispAmp = 0.10          // displacement amplitude
)
{
        point PP = point "shader" P;
        float magnitude = 0.;
        float size = 1.0;
        float i;
        for( i = 0.; i < 6.0; i += 1.0 )
        {
                magnitude += (noise(size * PP) - 0.5) / size;
                size *= 2.0;
        }

        float Ar = Ad / 2.0;
        float Br = Bd / 2.0;

        float up = 2. * u;
        float vp = v;

        float numinu = floor( up / Ad );
        float numinv = floor( vp / Bd );

        float uc = numinu * Ad + Ar;
        float vc = numinv * Bd + Br;

        float du = up - uc;
        float dv = vp - vc;

        float factor = 4.0 * magnitude;

        float d = (((0.0 - du) / Ar) * ((0.0 - du) / Ar)) + (((0.0 - dv) / Br) * ((0.0 - dv) / Br));

        if( d + factor <= 1.0 )
        {
                P = P + ((1.0 - (d + factor)) * DispAmp * normalize(N));
        }

        N = calculatenormal(P);
}
```

noisy_bumped_sphere.rib

```
##RenderMan RIB
version 3.03

Declare "Ad"    "uniform float"
Declare "Bd"    "uniform float"
Declare "NoiseAmp"   "uniform float"
Declare "DispAmp"     "uniform float"

Display "noisy_bumped_sphere.tiff" "file" "rgb"
Format 1024 1024 -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
        Displacement "noisy_bumped_sphere" "Ad" 0.100 "Bd" 0.050
        Surface "noisy_sphere" "Ad" 0.100 "Bd" 0.050
        Color [0.2 0.2 1]
        Opacity [1 1 1]
        TransformBegin
                Rotate 90 1. 0. 0.
                Sphere 3 -3 3 360
        TransformEnd
WorldEnd
```

noisy_bumped_sphere.sl

```
displacement
noisy_bumped_sphere( float
            Ad = 0.100,             // u diameter
            Bd = 0.050,              // v diameter
            DispAmp = 0.10          // displacement amplitude
)
{
        point PP = point "shader" P;
        float magnitude = 0.;
        float size = 1.0;
        float i;
        for( i = 0.; i < 6.0; i += 1.0 )
        {
                magnitude += (noise(size * PP) - 0.5) / size;
                size *= 2.0;
        }

        float Ar = Ad / 2.0;
        float Br = Bd / 2.0;

        float up = 2. * u;
        float vp = v;

        float numinu = floor( up / Ad );
        float numinv = floor( vp / Bd );

        float uc = numinu * Ad + Ar;
        float vc = numinv * Bd + Br;

        float du = up - uc;
        float dv = vp - vc;

        float factor = 4.0 * magnitude;

        float d = (((0.0 - du) / Ar) * ((0.0 - du) / Ar)) + (((0.0 - dv) / Br) * ((0.0 - dv) / Br));

        if( d + factor <= 1.0 )
        {
                N = calculatenormal(P + ((1.0 - (d + factor)) * DispAmp * normalize(N)));
        } else {
                N = calculatenormal(P);
        }
}
```

Deconstruction

I began with my previous submission for project one, modifying it with the provided noise generation. It contains six octaves of noise, which produces relatively “hilly” noise, with large slopes and minor slopes within. Using the produced noise I used it to modify the distance algorithm, allowing points that were technically inside or outside of the original ellipse to be respectively changed. This made it so points there were inside before, now fall outside, and points outside can now fall within. This created the first image shown, in which you can see the rough outline of the original ellipses. Branching from this I created the displacement shader and implemented the same algorithm to produce identical noise and used the same distance gate as before, but instead I had to invert the distance, multiply it by the magnitude of the noise, and lastly multiply it by the normal of the surface. With some scaling it produces lifted segments from the surface of the sphere, in which the points closest to the center, modified by the fake input of noise, are the tallest points. The points closest to the edge of the ellipse are the lowest points, to properly blend into the original structure of the sphere. Lastly I modified this to produce the bump mapped sphere, which I simply stopped modifying the surface position, and instead modified the surface normal alone. This gives the illusion of the surface having “noisy” areas, but in reality is simply a round sphere that reflects light differently due to the different surface normals.

Images