Project 1

Alan Neads

24th January, 2017

elliptical_dots.rib

```
##RenderMan RIB
version 3.03

Declare "Ad"	"uniform float"
Declare "Bd"	"uniform float"

Display "elliptical_dots.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
	Surface "ovals" "Ad" 0.020 "Bd" 0.050
	Color [1 1 1]
	Opacity [1 1 1]
	TransformBegin
		Rotate 90 1. 0. 0.
		Sphere 3 -3 3 360
	TransformEnd
WorldEnd
```

ovals.sl

```
surface
ovals( float
                        Ad = 0.0025, // dot radius A
                        Bd = 0.10, // dot radius B
                        Ks = 0.5,
                        Kd = 0.5,
                        Ka = .1,
                        roughness = 0.1;
        color   specularColor = color( 1, 1, 1 )
)
{
        float up = 2. * u;
        float vp = v;
        float numinu = floor( up / (2. * Ad) );
        float numinv = floor( vp / (2. * Bd) );

        color dotColor = Cs;
        if( mod( numinu+numinv, 2 ) == 0 )
        {
                float uc = numinu * 2. * Ad + Ad;
                float vc = numinv * 2. * Bd + Bd;
                up = up - uc;
                vp = vp - vc;
                float distA = (0. - up) / Ad;
                float distB = (0. - vp) / Bd;
                distA = distA * distA;
                distB = distB * distB;
                if( distA + distB <= 1 )
                {
                        dotColor = color( 1., .5, 0. ); // beaver orange?
                }
        }
        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 ) );
}
```

Deconstruction

I began with the basic dots.rib and dots.sl files, which produces an output of orange dots on a white sphere. The dots have a predetermined diameter specified with the Diam variable. Modifying the value on the shader line will increase or decrease the size of the dots respectively. In order to modify this we need to replace the Diam variable with two variables, Ad and Bd. Each variable represents one of the axes that creates an ellipse. Therefore the default values of Ab and Bd must reflect this, with 0.0025 for Ab and 0.10 for Bd. This creates a skewed rectangle in which the ellipse will be calculated and shaded. To determine whether the pixel is within the ellipse or not we must break down the location of the pixel within the already defined rectangle. We can determine whether or not the pixels location falls within the ellipse or it's containing rectangle by the formula:

$$\left(\frac{u-u_c}{A_r}\right)^2 + \left(\frac{v-v_c}{B_r}\right)^2 \leq 1$$

This formula essentially takes the center of the ellipse (u,v), the pixels location (u_c,v_c), the major and minor axes radius (A_r,B_r), and if the equation falls true when all of the values are input, it indicates the pixels location falls within the confines of the ellipse. If the left hand side of the equation however produces a result greater than 1, it falls outside of the ellipse. For our instance we want to shade the interior of the ellipse orange, but the exterior will remain the predetermined white.

Images