```
//
// SAMPLE BOWLING PIN SURFACE SHADER
//
// Based on the RenderMan bowling pin example
//

// Useful constant

constant floatv Zero = { 0, 0, 0, 0 };

// Reflection functions

surface floatv
lightmodel_diffuse (floatv a, floatv d)
{
    perlight float diffuse = dot(N,L);
    perlight floatv fr = select(diffuse > 0, d * diffuse, Zero);
    return a * Ca + integrate(fr * Cl);
}

surface floatv
lightmodel_specular (floatv s, floatv e, float sh)
{
    perlight float diffuse = dot(N,L);
    perlight float specular = pow(max(dot(N,H),0),sh);
    perlight floatv fr = select(diffuse > 0, s * specular, Zero);
    return integrate(fr * Cl) + e;
}

surface shader floatv
bowling_pin (texref pinbase, texref bruns, texref circle, texref coated,
             texref marks, floatv uv)
{
    floatv uv_wrap = { uv[0], 10 * Pobj[1], 0, 1 };
    floatv uv_label = { 10 * Pobj[0], 10 * Pobj[1], 0, 1 };
    matrix t_base = invert(translate(0, -7.5, 0) * scale(0.667, 15, 1));
    matrix t_bruns = invert(translate(-2.6, -2.8, 0) * scale(5.2, 5.2, 1));
    matrix t_circle = invert(translate(-0.8, -1.15, 0) * scale(1.4, 1.4, 1));
    matrix t_coated = invert(translate(2.6, -2.8, 0) * scale(-5.2, 5.2, 1));
    matrix t_marks = invert(translate(2.0, 7.5, 0) * scale (4, -15, 1));
    float front = select(Pobj[2] >= 0, 1, 0);
    float back = select(Pobj[2] <= 0, 1, 0);
    floatv Base = texture(pinbase, t_base * uv_wrap);
    floatv Bruns = front * texture(bruns, t_bruns * uv_label);
    floatv Circle = front * texture(circle, t_circle * uv_label);
    floatv Coated = back * texture(coated, t_coated * uv_label);
    floatv Marks = texture(marks, t_marks * uv_wrap);
    floatv Cd = lightmodel_diffuse({ 0.4, 0.4, 0.4, 1 }, { 0.5, 0.5, 0.5, 1 });
    floatv Cs = lightmodel_specular({ 0.35, 0.35, 0.35, 1 }, Zero, 20);
    return (Circle over (Bruns over (Coated over Base))) * (Marks * Cd) + Cs;
}
```

```
//
// SAMPLE LIGHT SHADERS
//


// Attenuation function

light float
atten (float ac, float al, float aq)
{
    return 1.0 / ((aq * Sdist + al) * Sdist + ac);
}

// A simple point light shader

light shader floatv
simple_light (floatv color, float ac, float al, float aq)
{
    return color * atten(ac, al, aq);
}

// A spotlight shader with associated functions

float
smoothstep (float value, float min, float max)
{
    float t = clamp((value - min) / (max - min), 0, 1);
    return t * t * (3 - 2 * t);
}

float
smoothspot (float spot_cos, float inner_edge_angle, float outer_edge_angle)
{
    float inner_cos = cos(inner_edge_angle * pi / 180);
    float outer_cos = cos(outer_edge_angle * pi / 180);
    return smoothstep(spot_cos, outer_cos, inner_cos);
}

light shader floatv
spotlight (floatv color, float ac, float al, float aq)
{
    floatv Cl = smoothspot(-S[2], 15, 30) * color * atten(ac, al, aq);
    return Cl;
}

// A spotlight with a projective slide image

light shader floatv
star_projector (floatv color, float ac, float al, float aq, texref stars)
{
    float time = fixedtime;
    floatv Cl = smoothspot(-S[2], 15, 30) * color * atten(ac, al, aq);
    floatv uv = { S[0], S[1], 0, -S[2] }; // project
    return Cl * texture(stars, scale(1.5, 1.5, 1) * uv);
}
```