Rendering Techniques Handout — Noise-based Shaders

Star Wallpaper

```
#define NCELLS 10
#define CELLSIZE (1/NCELLS)
#define snoise(s,t)       (2*noise((s),(t))-1)

surface
wallpaper(
   uniform float Ka = 1;
   uniform float Kd = 1;
   uniform color starcolor = color (1.0000,0.5161,0.0000);
   uniform float npoints = 5;
    )
{
   color Ct;
   point Nf;
   float ss, tt, angle, r, a, in_out;
   float sctr, tctr, scell, tcell;
   float scellctr, tcellctr;
   float i, j;
   uniform float rmin = 0.01, rmax = 0.03;
   uniform float starangle = 2*PI/npoints;
   uniform point p0 = rmax*(cos(0),sin(0),0);
   uniform point p1 = rmin*
      (cos(starangle/2),sin(starangle/2),0);
   uniform point d0 = p1 - p0;
   point d1;

   scellctr = floor(s*NCELLS);
   tcellctr = floor(t*NCELLS);
   in_out = 0;

   for (i = -1; i <= 1; i += 1) {
      for (j = -1; j <= 1; j += 1) {
            scell = scellctr + i;
            tcell = tcellctr + j;
            if (float noise(3*scell-9.5,7*tcell+7.5) < 0.55) {
            sctr = CELLSIZE * (scell + 0.5
                + 0.6 * snoise(scell+0.5, tcell+0.5));
            tctr = CELLSIZE * (tcell + 0.5
                + 0.6 * snoise(scell+3.5, tcell+8.5));
            ss = s - sctr;
            tt = t - tctr;
```

```
            angle = atan(ss, tt) + PI;
            r = sqrt(ss*ss + tt*tt);
            a = mod(angle, starangle)/starangle;

            if (a >= 0.5)
               a = 1 - a;
            d1 = r*(cos(a), sin(a),0) - p0;
            in_out += step(0, zcomp(d0^d1));
         }
      }
   }
   Ct = mix(Cs, starcolor, step(0.5,in_out));

   /* "matte" reflection model */
   Nf = normalize(faceforward(N, I));
   Oi = Os;
   Ci = Os * Ct * (Ka * ambient() + Kd * diffuse(Nf));
}
```

Perturbed Texture

```
#include "proctext.h"

surface
perturb ()
{
   point Psh;
   float ss, tt;

   Psh = transform("shader", P) * 0.2;
   ss = s + 0.1 * snoise(Psh);
   tt = t + 0.05 * snoise(Psh+(1.5,6.7,3.4));
   Ci = texture("example.tx", ss, tt);
}
```

Blue Marble

```
#include "proctext.h"

#define PALE_BLUE        color (0.25, 0.25, 0.35)
#define MEDIUM_BLUE      color (0.10, 0.10, 0.30)
#define DARK_BLUE        color (0.05, 0.05, 0.26)
#define DARKER_BLUE      color (0.03, 0.03, 0.20)
#define NNOISE           4

color
marble_color(float m)
{
   return color spline(
      clamp(2*m + .75, 0, 1),
      PALE_BLUE, PALE_BLUE,
      MEDIUM_BLUE, MEDIUM_BLUE, MEDIUM_BLUE,
      PALE_BLUE, PALE_BLUE,
      DARK_BLUE, DARK_BLUE,
      DARKER_BLUE, DARKER_BLUE,
      PALE_BLUE, DARKER_BLUE);
}

surface
blue_marble(
   uniform float Ka = 1;
   uniform float Kd = 0.8;
   uniform float Ks = 0.2;
   uniform float texturescale = 2.5;
   uniform float roughness = 0.1;
    )
{
   color Ct;
   point NN;
   point PP;
   float i, f, marble;

   NN = normalize(faceforward(N, I));
   PP = transform("shader", P) * texturescale;

   marble = 0; f = 1;
   for (i = 0; i < NNOISE; i += 1) {
      marble += snoise(PP * f)/f;
      f *= 2.17;
   }
```

```
    Ct = marble_color(marble);

    Ci = Os * (Ct * (Ka * ambient() + Kd * diffuse(NN))
        + Ks * specular(NN, normalize(-I), roughness));
}
```