

```
1 // 被写界深度
2 //
3 string XFile = "misc/nanase_pose.x";
4 int BCLR = 0x000000ff;
5
6 // 変換マトリックス
7 float4x4 matWVP : WORLDVIEWPROJECTION; // ワールド×ビュー×射影
8 float4x4 matW : WORLD; // ワールド
9
10 float3 camera : CAMERAPOSITION; // 視点座標
11
12 // 光源ベクトル (光の進む方向)
13 float3 light <string UIDirectional = "Light Direction";> = {-0.35f, -0.25f, 0.9f};
14
15 // 光源の色
16 float4 I_d = {1.0f, 1.0f, 1.0f, 1.0f}; // 拡散光
17 float4 I_a = {0.2f, 0.2f, 0.2f, 1.0f}; // 環境光
18 float4 I_s = {1.0f, 1.0f, 1.0f, 1.0f}; // 鏡面反射光
19
20 // Xファイル マテリアル
21 float4 k_d : MATERIALDIFFUSE; // 拡散色
22 float4 k_s : MATERIALSPECULAR; // 鏡面反射色
23 float k_p : MATERIALPOWER; // 鏡面反射指数
24 float4 k_a : MATERIALAMBIENT; // 環境色
25 texture tex : MATERIALTEXTURE;
26 bool tex_enable : MATERIALTEXTUREVALID;
27
28 texture texBlur; // ぼかしテクスチャ
29 texture texDepth; // 深度テクスチャ
30 float focus; // ピント位置(0~1)
31 float range = 10.0f; // フォーカス範囲
32
33 // サンプラステート
34 sampler Sampler = sampler_state
35 {
36     Texture = (tex);
37     MipFilter = LINEAR;
38     MinFilter = LINEAR;
39     MagFilter = LINEAR;
40 };
41 sampler SamplerBlur = sampler_state
42 {
43     Texture = (texBlur);
44     MipFilter = LINEAR;
45     MinFilter = LINEAR;
46     MagFilter = LINEAR;
47 };
48 sampler SamplerDepth = sampler_state
49 {
50     Texture = (texDepth);
51     MipFilter = POINT;
52     MinFilter = POINT;
53     MagFilter = POINT;
54 };
55
56 // 頂点シェーダ
57 void PhongVS(float4 InPos : POSITION,
58             float3 InNor : NORMAL,
59             float2 InTex : TEXCOORD0,
60             out float4 OutPos : POSITION,
61             out float2 OutTex : TEXCOORD0,
62             out float3 OutWPos : TEXCOORD1,
63             out float3 OutWNor : TEXCOORD2,
64             out float4 OutDpth : TEXCOORD3)
65 {
66     OutPos = mul(InPos, matWVP);
67     OutTex = InTex;
68     OutWPos = mul(InPos, matW).xyz;
69     OutWNor = mul(InNor, (float3x3)matW);
70     OutDpth = OutPos; // Z値出力用
71 }
72
73 // ピクセルシェーダ
```

```

74 void PhongPS(float2 InTex : TEXCOORD0,
75             float3 InPos : TEXCOORD1,
76             float3 InNor : TEXCOORD2,
77             float4 InDpth : TEXCOORD3,
78             out float4 OutCol : COLOR0,
79             out float4 OutDpth : COLOR1)
80 {
81     float z = InDpth.z / InDpth.w;
82     OutDpth = float4(z, z, z, 1);
83
84     float3 L = normalize(-light);
85     float3 N = normalize(InNor);
86     float3 V = normalize(camera - InPos);
87     float3 R = reflect(-V, N);
88     OutCol = I_a * k_a + I_d * k_d * saturate(dot(L, N));
89     if (tex_enable) {
90         OutCol *= tex2D(Sampler, InTex);
91     }
92     OutCol += I_s * k_s * pow(saturate(dot(L, R)), k_p * 0.25f);
93 }
94
95 // 頂点シェーダ(単純コピー)
96 void SimpleVS(float4 InPos : POSITION,
97             float2 InTex : TEXCOORD0,
98             out float4 OutPos : POSITION,
99             out float2 OutTex : TEXCOORD0)
100 {
101     OutPos = InPos;
102     OutTex = InTex;
103 }
104
105 // ピクセルシェーダ(強烈ぼかし&1/2コピー)
106 void DiffusePS(float2 InTex : TEXCOORD0,
107             out float4 OutCol : COLOR0)
108 {
109     // テクセル中央にずらす(出力サイズ400x300時)
110     InTex.x += 0.5f / 400;
111     InTex.y += 0.5f / 300;
112     // テクセルサイズ計算
113     float dx = 1.0f / 400;
114     float dy = 1.0f / 300;
115     // 該当テクセル周辺の平均を求める(9x9)
116     OutCol = float4(0, 0, 0, 0);
117     for (int y = -4; y < 5; ++y) {
118         for (int x = -4; x < 5; ++x) {
119             OutCol += tex2D(Sampler,
120                 InTex + float2(dx * x, dy * y)) / 81.0f;
121         }
122     }
123     // アルファ値補正
124     OutCol.a = 1.0f;
125 }
126
127 // 被写界深度
128 void FocusPS(float2 InTex : TEXCOORD0,
129             out float4 OutCol : COLOR0)
130 {
131     float depth = tex2D(SamplerDepth, InTex).r;
132     float blend = abs(depth - focus) * range;
133     blend = saturate(blend);
134
135     OutCol = tex2D(Sampler, InTex) * (1.0f - blend)
136         + tex2D(SamplerBlur, InTex) * blend;
137 }
138
139 // 通常描画
140 technique TPhong
141 {
142     // パス
143     pass P0
144     {
145         VertexShader = compile vs_3_0 PhongVS();
146         PixelShader = compile ps_3_0 PhongPS();

```

```
147     }
148 }
149
150 // ぼかしシェーダ
151 technique TDiffuse
152 {
153     pass P0
154     {
155         VertexShader = compile vs_3_0 SimpleVS();
156         PixelShader = compile ps_3_0 DiffusePS();
157     }
158 }
159
160 // 被写界深度
161 technique TFocusFilter
162 {
163     pass P0
164     {
165         VertexShader = compile vs_3_0 SimpleVS();
166         PixelShader = compile ps_3_0 FocusPS();
167     }
168 }
```