

リスト 6.7	reflectBump.frag
<pre> <b>varying vec3</b> P; //位置ベクトル <b>varying vec3</b> N; //法線ベクトル <b>varying vec3</b> T; //接線ベクトル <b>varying vec3</b> B; //従法線ベクトル <b>uniform mat4</b> ViewTranspose; //ビュー変換だけのモデル・ビュー行列 <b>uniform float</b> reflectivity; //反射率 <b>uniform samplerCube</b> cubeMap; //環境マップ <b>uniform sampler2D</b> normalMap; //法線マップ  <b>void main(void)</b> {     <b>vec3</b> grad = (texture2D(normalMap, gl_TexCoord[0].st).xyz - 0.5) * 2.0;     N = normalize( N - grad.x * T - grad.y * B);     //反射ベクトルの計算     <b>vec3</b> incident = normalize(P); //入射視線ベクトル     <b>vec3</b> reflect0 = reflect(incident, N); //その反射ベクトル     <b>vec4</b> Reflect = ViewTranspose * <b>vec4</b>(reflect0, 0.0);     //以下は反射環境マッピング     <b>vec3</b> L = normalize(gl_LightSource[0].position.xyz - P);     <b>vec4</b> ambient = gl_FrontLightProduct[0].ambient;     <b>float</b> dotNL = dot(N, L);     <b>vec4</b> diffuse = gl_FrontLightProduct[0].diffuse * max(0.0, dotNL);     <b>vec3</b> V = normalize(-P);     <b>vec3</b> H = normalize(L + V);     <b>float</b> powNH = pow(max(dot(N, H), 0.0), gl_FrontMaterial.shininess);     <b>if</b>(dotNL &lt;= 0.0) powNH = 0.0;     <b>vec4</b> specular = gl_FrontLightProduct[0].specular * powNH;     //オブジェクトの色と環境マップを線形補間     gl_FragColor = mix(ambient + diffuse, textureCube(cubeMap, Reflect.stp), reflectivity) + specular; } </pre>	