リスト9.6 bubble.frag

```
varying vec3 P;
varying vec3 N;
varying vec4 Reflect;//反射ベクトル
uniform samplerCube envSampler;
uniform float alpha;

uniform float refract;//相対屈折率
uniform float filmSpace ;//nano meter

const float ultraviolet = 380.0;
const float infrared = 780.0;

vec3 getRGB(float lambda)
{
 //中略(リスト9.4に同じ
}

void main(void)
{
  vec3 L = normalize(gl_LightSource[0].position.xyz - P);
  N = normalize(N);

  vec3 V = normalize(-P);
  vec3 H = normalize(L + V);
  float powNH1 = pow(max(dot(N, H), 0.0), gl_FrontMaterial.shininess);
  float powNH2 = pow(max(dot(-N, H), 0.0), gl_FrontMaterial.shininess);
  vec4 specular = gl_FrontLightProduct[0].specular * powNH1;
  specular += gl_FrontLightProduct[0].specular * powNH2;

  //干渉による色の計算
  float dotNV = dot(N, V);
  float cosThetaR = sqrt(1.0 - (1.0 - dotNV*dotNV)/(refract * refract));
  float a = 2.0 * filmSpace * cosThetaR * refract;
  vec4 colorInter = vec4(0.0);
  for(int i = 0; i <= 5; i++)
  {
    float lambda = a / (float(i) + 0.5);
    if(lambda < ultraviolet || lambda > infrared) continue;
    colorInter.rgb += getRGB(lambda);
  }

  vec4 colorReflect = textureCube(envSampler, Reflect.stp) ;//反射環境の色
  vec3 col = (colorReflect+ specular).rgb;
  if(colorInter.rgb == vec3(0.0) ) gl_FragColor.rgb = col;
  else                             gl_FragColor.rgb = colorInter.rgb * col;

  col = col*col*col*col;
  gl_FragColor.a = (col.r + col.g + col.b) * alpha;
}
```