Unified Distance Formulas for Halfspace Fog

Eric Lengyel
Terathon Software
lengyel@terathon.com

Abstract. In many real-time rendering applications, it is necessary to model a fog volume that is bounded by a single plane but is otherwise infinite in extent. This paper presents unified formulas that provide the correct distance traveled through a fog halfspace for all possible camera and surface point locations. Such formulas effectively remove the need to code for multiple cases separately, thereby achieving optimal fragment shading performance on modern rendering hardware.

1. Introduction

A fog volume bounded by a single plane, but otherwise infinite in extent, has many uses in interactive rendering applications. For example, the bounding plane could represent the maximum height that a fog bank reaches in an outdoor environment, it could coincide with a water plane to restrict fog application to underwater surfaces, or it could serve as the boundary between fogged and unfogged indoor areas. In such cases, the partial distance within the fog volume through which light travels between a surface point and the camera must be determined for each pixel rendered. Managing this calculation can be cumbersome, however, due to the different possible configurations of a surface point and the camera position with respect to the bounding plane. Furthermore, an implementation that handles distinct configurations as separate cases can suffer from poor performance caused by fragment shader code containing branches or using some kind of conditional execution. We therefore derive unified formulas that provide the correct distance traveled through a fog halfspace for all surface points and camera positions, allowing a single fragment shader to be used in all cases. One formula is derived for a volume having a constant fog density, and a second formula is derived for a volume having a fog density that increases linearly with distance from the bounding plane.

2. Background

As summarized in the following table, OpenGL defines three different fog modes that each calculate a fog fraction f in terms of a fog density ρ and the distance d that light

travels through the fog [Leech 04]. (In the `GL_LINEAR` case, an additional constant *e* represents the distance at which the fog fraction is zero, corresponding to fully saturated fog.)

Fog Mode	Fog Fraction
`GL_LINEAR`	$f = \rho(e-d)$
`GL_EXP`	$f = \exp(-\rho d)$
`GL_EXP2`	$f = \exp\left(-\rho^2 d^2\right)$

Once *f* has been calculated for a fragment being rasterized, it is clamped to the range $[0,1]$, and the fragment's final color C_r is blended with a constant fog color C_f to produce the color *C* which is written to the frame buffer using the formula

$$C = f C_r + (1-f) C_f. \tag{1}$$

Ordinarily, the distance *d* is taken to be the total distance between the camera and the center of a fragment being rasterized.[1] In this paper, we consider fog volumes that are bounded by a plane dividing space into a fogged halfspace and an unfogged halfspace. To calculate the appropriate fog fraction *f* in this case for an arbitrary camera position, we need to determine the portion of the distance between the camera and a fragment's center that lies within the fogged halfspace to use as the value of *d*. Note that once the correct distance *d* has been calculated, any formula may be used to calculate the fog fraction *f*, not just the conventional OpenGL formulas.

We represent the bounding plane between fogged and unfogged halfspaces by the four-dimensional vector $\mathbf{F} = \langle F_x, F_y, F_z, F_w \rangle$. We shall assume that the plane's normal (described by the *x*, *y*, and *z* components of **F**) has unit length and, as a convention, that the normal points away from the fogged halfspace. Recall that the dot product between the plane **F** and a homogeneous point **P** having a *w* coordinate of one gives the signed distance between the plane and the point. As illustrated in Figure 1, all points having a negative distance to the plane lie in the fogged halfspace, and all points having a positive distance to the plane lie in the unfogged halfspace. As explained later, whether points lying in the plane itself are considered fogged is unimportant and may be chosen one way or the other.

[1]The OpenGL specification allows an implementation to approximate this distance with the fragment's camera-space depth.

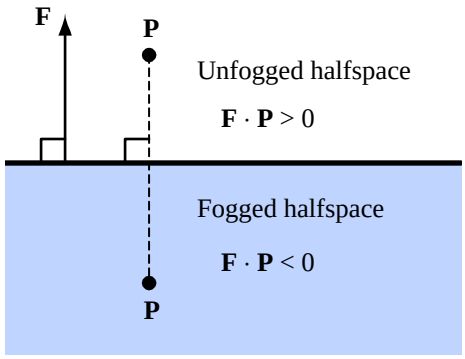

Figure 1. A point **P** in the fogged halfspace forms a negative dot product with the plane vector **F**, and a point in the unfogged halfspace forms a positive dot product with the plane vector.

3. Constant Density Fog Halfspace

We first consider a volume in which the fog density is given by a constant ρ. Suppose that the point **P** represents a fragment inside a triangle being rasterized and that the point **C** represents the camera position. When determining the partial distance along the direction from **P** to **C** lying within the fogged halfspace, there are four cases to consider based on the signs of $\mathbf{F} \cdot \mathbf{P}$ and $\mathbf{F} \cdot \mathbf{C}$, as shown in Figure 2.

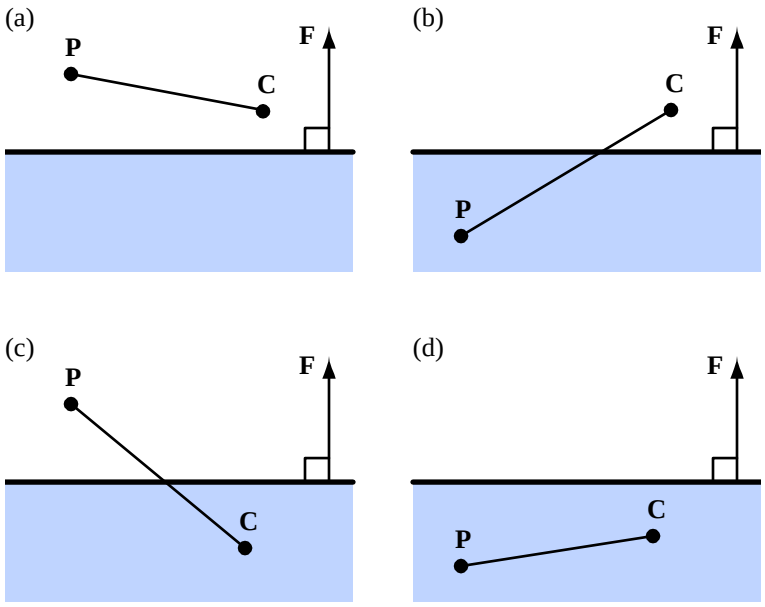

Figure 2. There are four distinct configurations to consider when calculating the partial distance along the direction from a surface point **P** to the camera position **C** that lies within the fogged halfspace.

In the case that **P** and **C** both lie on the positive side of the fog plane, no part of the line segment connecting the two points intersects the fogged halfspace, and thus no fog should be applied to the fragment corresponding to the point **P**. In the case that **P** and **C** both lie on the negative side of the fog plane, the entire distance $\|\mathbf{C}-\mathbf{P}\|$ should be used to calculate the amount of fog that should be applied.

The remaining two cases, in which $\mathbf{F}\cdot\mathbf{P}$ and $\mathbf{F}\cdot\mathbf{C}$ have opposite signs, require us to calculate the point at which the line segment connecting **P** and **C** intersects the plane **F**. If we define the function $\mathbf{Q}(t)=\mathbf{P}+t\mathbf{V}$, where we have introduced $\mathbf{V}=\mathbf{C}-\mathbf{P}$ as the traditional "view vector", then a quick calculation yields

$$t=-\frac{\mathbf{F}\cdot\mathbf{P}}{\mathbf{F}\cdot\mathbf{V}} \tag{2}$$

as the parameter value for which $\mathbf{Q}(t)$ lies in the plane. (Note that the *w* coordinate of **V** is zero.) In the case that $\mathbf{F}\cdot\mathbf{C}>0$, the distance traveled through the fog between **P** and **C** is $t\|\mathbf{V}\|$. In the case that $\mathbf{F}\cdot\mathbf{C}<0$, the fogged distance is $(1-t)\|\mathbf{V}\|$.

The following table summarizes the distances traveled through the fogged halfspace for the four cases illustrated in Figure 2.

Case	$\mathbf{F}\cdot\mathbf{P}$	$\mathbf{F}\cdot\mathbf{C}$	Distance *d*
(a)	Positive	Positive	0
(b)	Negative	Positive	$-\frac{\mathbf{F}\cdot\mathbf{P}}{\mathbf{F}\cdot\mathbf{V}}\|\mathbf{V}\|$
(c)	Positive	Negative	$\left(1+\frac{\mathbf{F}\cdot\mathbf{P}}{\mathbf{F}\cdot\mathbf{V}}\right)\|\mathbf{V}\|$
(d)	Negative	Negative	$\|\mathbf{V}\|$

To satisfy the goal of optimal real-time rendering performance, we seek a single formula that gives the correct distance *d* for all cases simultaneously. Such a formula avoids expensive branches or other conditional code in the fragment shader.

First, we can make use of the fact that $\mathbf{F}\cdot\mathbf{V}$ is always positive in case (b) and always negative in case (c). By applying an absolute value to $\mathbf{F}\cdot\mathbf{V}$, we can merge cases (b) and (c) into one formula and write *d* as

$$d=\left(k-\frac{\mathbf{F}\cdot\mathbf{P}}{|\mathbf{F}\cdot\mathbf{V}|}\right)\|\mathbf{V}\|, \tag{3}$$

where the constant *k* is defined as

$$k=\begin{cases}1, & \text{if } \mathbf{F}\cdot\mathbf{C}\le 0;\\ 0, & \text{otherwise.}\end{cases} \tag{4}$$

If Equation (3) were applied to case (a), then we would always obtain a negative distance when we want a distance of zero. If it were applied to case (d), then we would always obtain a distance greater than $\|\mathbf{V}\|$ when we want the distance $\|\mathbf{V}\|$ itself. Fortunately, this means that we can clamp the scale value to the range $[0,1]$ as follows to obtain the correct distance in all four cases.

$$d = \text{sat}\left(k - \frac{\mathbf{F}\cdot\mathbf{P}}{|\mathbf{F}\cdot\mathbf{V}|}\right)\|\mathbf{V}\| \tag{5}$$

The sat function represents the saturate operation available on all modern graphics processors that clamps values to the range $[0,1]$.

If the direction between the points **P** and **C** is perpendicular to the fog boundary plane's normal, then $\mathbf{F}\cdot\mathbf{V} = 0$ and we encounter a division-by-zero hazard in Equation (5). In this case, $\mathbf{F}\cdot\mathbf{P}$ and $\mathbf{F}\cdot\mathbf{C}$ must have the same sign, so the distance d becomes

$$d = \text{sat}(-\text{sgn}(\mathbf{F}\cdot\mathbf{C})\infty)\|\mathbf{V}\|. \tag{6}$$

The saturate operation maps the infinity to one if the camera lies inside the fog volume and zero otherwise, giving the correct results.

If the points **P** and **C** both lie in the plane **F**, then $\mathbf{F}\cdot\mathbf{P} = 0$ and $\mathbf{F}\cdot\mathbf{V} = 0$, leading to a zero-divided-by-zero expression in Equation (5) that produces a floating-point NaN (Not a Number) value. Whether a NaN is mapped to zero or one by the saturate operation is unspecified, but either case is acceptable because the pixel corresponding to the point **P** is guaranteed to have neighbors for which $d = 0$ and neighbors for which $d = \|\mathbf{V}\|$.

4. Linear Density Fog Halfspace

Multiplying the distance given by Equation (5) by a constant density value produces a fog volume that can appear too thick near the boundary plane. A constant density can also produce the appearance of a harsh transition between fogged and unfogged pixels when the camera is positioned near the boundary plane. A more natural result can be achieved by utilizing a density function $\rho(\mathbf{P})$ that is zero on the bounding plane and increases linearly with distance into the fog volume. A comparison between constant density and linear density is shown in Figure 3.

Let the density function $\rho(\mathbf{P})$ be defined as

$$\rho(\mathbf{P}) = -a(\mathbf{F}\cdot\mathbf{P}) \tag{7}$$

for some positive constant a, and let dg represent the amount of fog applied along a differential length ds, given by the product

$$dg = \rho(\mathbf{P})\,ds. \tag{8}$$

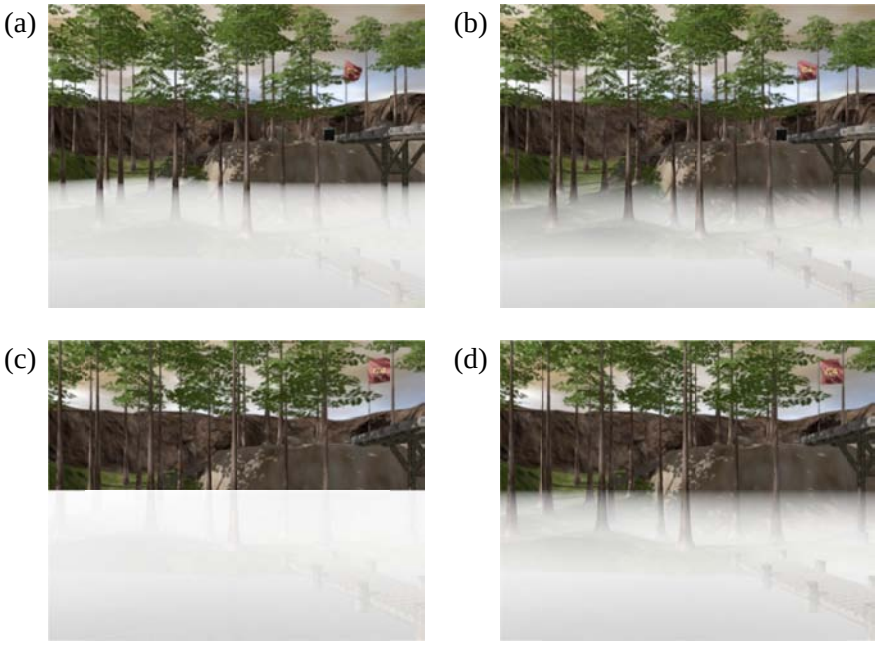

Figure 3. Images (a) and (c) show a fog volume rendered with constant density $\rho = 0.04$, and images (b) and (d) show a fog volume rendered with a linear density function $\rho(\mathbf{P}) = -0.008(\mathbf{F} \cdot \mathbf{P})$. The camera is placed 5 meters above the fog volume's boundary plane in images (a) and (b), and the camera lies in the boundary plane in images (c) and (d).

By integrating along the partial path between the points **P** and **C** that lies inside the fog volume, we obtain a function $g(\mathbf{P})$ that can be substituted for the product ρd in any of the fog fraction formulas.

Let the function $\mathbf{Q}(u) = \mathbf{P} + u\mathbf{V}$ represent the line segment connecting **P** and **C** with $u \in [0,1]$, where we continue to use $\mathbf{V} = \mathbf{C} - \mathbf{P}$. The differential distance ds can be expressed in terms of u as $ds = \|\mathbf{V}\| du$. We again need to consider the four possible configurations of the points **P** and **C** with respect to the boundary plane. Of course, if $\mathbf{F} \cdot \mathbf{P}$ and $\mathbf{F} \cdot \mathbf{C}$ are both positive, then $g(\mathbf{P}) = 0$. In the case that $\mathbf{F} \cdot \mathbf{P} < 0$ and $\mathbf{F} \cdot \mathbf{C} < 0$, we integrate over the entire distance between **P** and **C** to obtain

$$\begin{aligned} g(\mathbf{P}) &= \int_{\mathbf{P}}^{\mathbf{C}} dg = \int_0^1 \rho(\mathbf{Q}(u)) \|\mathbf{V}\| du \\ &= -\frac{a}{2} \|\mathbf{V}\| (\mathbf{F} \cdot \mathbf{P} + \mathbf{F} \cdot \mathbf{C}). \end{aligned} \tag{9}$$

In the remaining two cases, for which $\mathbf{F} \cdot \mathbf{P}$ and $\mathbf{F} \cdot \mathbf{C}$ have opposite signs, the parameter t given by Equation (2) is one of the limits of integration. If only $\mathbf{F} \cdot \mathbf{P} < 0$, we have

$$g(\mathbf{P}) = \int_0^t \rho(\mathbf{Q}(u)) \|\mathbf{V}\| du = \frac{a}{2} \|\mathbf{V}\| \frac{(\mathbf{F} \cdot \mathbf{P})^2}{\mathbf{F} \cdot \mathbf{V}}, \tag{10}$$

and if only $\mathbf{F}\cdot\mathbf{C}<0$, we have

$$g(\mathbf{P})=\int_t^1 \rho(\mathbf{Q}(u))\|\mathbf{V}\|du=-\frac{a}{2}\|\mathbf{V}\|\left(\mathbf{F}\cdot\mathbf{P}+\mathbf{F}\cdot\mathbf{C}+\frac{(\mathbf{F}\cdot\mathbf{P})^2}{\mathbf{F}\cdot\mathbf{V}}\right). \tag{11}$$

The fog functions $g(\mathbf{P})$ for all four cases shown in Figure 2 are summarized in the following table.

Case	$\mathbf{F}\cdot\mathbf{P}$	$\mathbf{F}\cdot\mathbf{C}$	Fog Function $g(\mathbf{P})$
(a)	Positive	Positive	0
(b)	Negative	Positive	$\frac{a}{2}\|\mathbf{V}\|\frac{(\mathbf{F}\cdot\mathbf{P})^2}{\mathbf{F}\cdot\mathbf{V}}$
(c)	Positive	Negative	$-\frac{a}{2}\|\mathbf{V}\|\left(\mathbf{F}\cdot\mathbf{P}+\mathbf{F}\cdot\mathbf{C}+\frac{(\mathbf{F}\cdot\mathbf{P})^2}{\mathbf{F}\cdot\mathbf{V}}\right)$
(d)	Negative	Negative	$-\frac{a}{2}\|\mathbf{V}\|(\mathbf{F}\cdot\mathbf{P}+\mathbf{F}\cdot\mathbf{C})$

As in the constant density formula, we can merge cases (b) and (c) by applying an absolute value to the quantity $\mathbf{F}\cdot\mathbf{V}$ and adjusting the sign of the term containing it. This yields the unified function

$$g(\mathbf{P})=-\frac{a}{2}\|\mathbf{V}\|\left[k(\mathbf{F}\cdot\mathbf{P}+\mathbf{F}\cdot\mathbf{C})-\frac{(\mathbf{F}\cdot\mathbf{P})^2}{|\mathbf{F}\cdot\mathbf{V}|}\right] \tag{12}$$

for these two cases, where k is still defined by Equation (4). In order to incorporate case (d) into this formula, we need to eliminate the last term inside the brackets whenever $\mathbf{F}\cdot\mathbf{P}$ and $\mathbf{F}\cdot\mathbf{C}$ have the same sign. This can be accomplished by replacing $\mathbf{F}\cdot\mathbf{P}$ with $\min((1-2k)(\mathbf{F}\cdot\mathbf{P}),0)$ in the last term to arrive at the formula

$$g(\mathbf{P})=-\frac{a}{2}\|\mathbf{V}\|\left[k(\mathbf{F}\cdot\mathbf{P}+\mathbf{F}\cdot\mathbf{C})-\frac{[\min((1-2k)(\mathbf{F}\cdot\mathbf{P}),0)]^2}{|\mathbf{F}\cdot\mathbf{V}|}\right]. \tag{13}$$

Note that this formula also works for case (a) since both terms are eliminated if $\mathbf{F}\cdot\mathbf{P}$ and $\mathbf{F}\cdot\mathbf{C}$ are both positive.

In Equation (13), if the quantity $\mathbf{F}\cdot\mathbf{V}$ is zero, then it is always true that the numerator of the last term is also zero. Although in practice a zero-divided-by-zero situation is rare enough to be ignored, a small positive ε can be added to $|\mathbf{F}\cdot\mathbf{V}|$ without affecting the fog function's value significantly to guarantee that a NaN is not produced.

5. Implementation

We finish with a discussion of the implementation of Equations (5) and (13) using OpenGL fragment shaders. Example code is given in the GL high-level shading language defined by the `GL_fragment_shader` extension, now a core feature of the OpenGL 2.0 standard.

For a fog volume of constant density, the quantities $\rho\mathbf{V}$, $\mathbf{F}\cdot\mathbf{P}$, and $\mathbf{F}\cdot\mathbf{V}$ are calculated in a vertex program and interpolated across triangles during rasterization. The value of k only needs to be calculated once for a particular camera position, and it is stored in a constant register. The following fragment shader can be used to calculate the product ρd and then apply the `GL_EXP` fog fraction formula. Note that to match the fog fraction that would be produced by OpenGL, the density ρ should be multiplied by $1/\ln 2$ because the `exp2` function exponentiates using the base 2 instead of the base e. (This function was chosen over the `exp` function to match the underlying hardware functionality of typical GPUs.)

```
uniform float k;            // (F dot C <= 0.0)
varying float3 rhoV;
varying float F_dot_V;
varying float F_dot_P;

void main()
{
    float4 color = ...;    // final color

    // Calculate distance * rho using Equation (5)
    float d = saturate(k - F_dot_P / abs(F_dot_V));
    d *= length(rhoV);

    // Calculate fog fraction and apply
    float f = saturate(exp2(-d));

    gl_FragColor.rgb = color.rgb * f +
        gl_FogParameters.color.rgb * (1.0 - f);
    gl_FragColor.a = color.a;
}
```

For a fog volume of linearly varying density, we can calculate the quantities $(a/2)\mathbf{V}$, $k(\mathbf{F}\cdot\mathbf{P}+\mathbf{F}\cdot\mathbf{C})$, and $(1-2k)(\mathbf{F}\cdot\mathbf{P})$ at each vertex and interpolate them during triangle rasterization. The fragment shader implementation of the linear density function is shown in the following code listing, which calculates $g(\mathbf{P})$ and then applies the `GL_EXP` fog fraction formula.

```
uniform float3 aV;          // (a / 2) * V
varying float c1;           // k * (F dot P + F dot C)
varying float c2;           // (1 - 2k) * (F dot P)
varying float F_dot_V;

void main()
{
    float4 color = ...;    // final color

    // Calculate g(P) using Equation (13)
    float g = min(c2, 0.0);
    g = -length(aV) * (c1 - g * g / abs(F_dot_V));

    // Calculate fog fraction and apply
    float f = saturate(exp2(-g));

    gl_FragColor.rgb = color.rgb * f +
        gl_FogParameters.color.rgb * (1.0 - f);
    gl_FragColor.a = color.a;
}
```

References

[Leech 04] Jon Leech and Pat Brown, eds. “The OpenGL Graphics System, Version 2.0.” Technical Specification, Silicon Graphics, Inc., 2004. Online at http://www.opengl.org/documentation/specs/version2.0/glspec20.pdf.

[Legakis 98] Justin Legakis. “Fast Multi-Layer Fog.” *Conference Abstracts and Applications (SIGGRAPH 98)*, p. 266, 1998.