ESライブラリ＆＆
ゲームプログラミング

○　バーテックスシェーダー編 - 第９回　バーテックスフォグ　○

バーテックスフォグ

・バーテックスフォグは、頂点と視点の距離に応じて霧がかかったようにぼやけさせる処理
・モデルが霧で包まれたような効果が得られる
・距離によって徐々に現れたり、消えるような表現ができる
・ただし、フォグ自体は(描画しない限り)見えない

概要

フォグ(霧)とは、モデルを視点との距離に応じて霧がかかったようにぼやけて見えるようにする効果のことです。

現実世界のフォグは、光が空気中の塵や水分で散乱され、その光が目に届くことで起こる大気効果です。物体から視点までの距離が長ければ長いほど、塵、水分の影響を多く受けるため、霧が濃くなったように感じます。

グラフィクスの世界では、フォグをかけることで、モデルを霧に包まれたようにしたり、遠くから徐々に現れるような効果を得ることができます。フォグ自体は描画されないので、フォグの模様を描いたテクスチャなどで再現します。

フォグの基本式

フォグを用いた場合、フォグの色とモデルの頂点色を合成します。もっとも基本的な式は、以下のものです。

出力色 = フォグの強さ * [頂点の色] + (1.0 - フォグの強さ) * [フォグの色]

フォグの強さを求める公式

フォグの強さを求める基本的な計算式として、「線形」「指数」「指数の２乗」の３つがあります。

線形公式は、以下の式で計算されます。始点と終点の間で距離に比例してフォグが強くなります。

(線形公式)

$$\text{フォグの強さ} = \frac{\text{end} - d}{\text{end} - \text{start}}$$

start = フォグ効果が始まる距離(開始点)
end = フォグ効果がこれ以上増加しなくなる距離(終了点)
d = 視点からオブジェクトまでの距離

２つの指数公式は、それぞれ以下の式で計算されます。

(指数公式)

$$\text{フォグの強さ} = \frac{1}{e^{(d \times \text{density})}}$$

(指数公式(２乗))

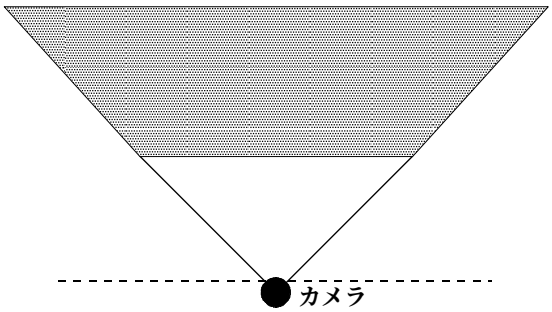
$$\text{フォグの強さ} = \frac{1}{e^{(d \times \text{density})^2}}$$

e = 自然対数(約2.71828)
d = 視点からオブジェクトまでの距離
density = フォグ密度(0.0～1.0の任意の値)

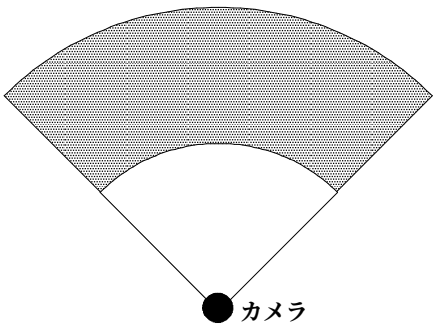
距離ベースのフォグ

通常、フォグを計算する場合、視点を平面(直線)に見立て、平面からの距離に比例させて平行的にフォグの強さを計算します。この方法は、視点の正面付近は問題ありませんが、横方向はフォグが掛かりません。視点を回転したとたん、横方向の頂点が正面になるため、急にフォグが掛かるといった不自然な状況が起こり得ます。より正確するには「距離ベース」で計算します。

距離をもとに計算する方法では、視点は点として扱い、点から頂点までの距離をフォグの計算に使用します。視点と頂点の２点間の距離の増加に従ってフォグ効果を増加させるため、視点の回転による不自然なフォグ効果を回避することができます。



▲通常の平行的なフォグ



▲距離ベースのフォグ

平行的なフォグが使われる理由は、計算にかかるコストが少ないためです。平行的なフォグでは、距離を求めるのに内積１回の計算で済みますが、距離ベースのフォグでは、ピタゴラスの定理を用いるため、２乗(または乗算)と足し算、さらに平方根の計算が必要です。頂点の数に比例して計算量も増加するため、それなりのコスト増になります。

高さベースのフォグ

高さベースのフォグは，視点からの距離ではなく、基準点からの高さによってフォグの強さを決めます。上から見下ろす場合、高さが低くなればなるほど視点からの距離が遠くなるため、「塵、水分の影響を多く受ける」ようになります。よって、霧が濃くなるようにします。フォグの強さを求める式は、「距離」が「高さ」になるだけで、まったく同じものが使用できます。

課　　題

バーテックスシェーダーでフォグを計算し、モデルにフォグ効果をつけましょう。

(１)バーテックスシェーダーでフォグを計算するため、Direct3Dのフォグ計算を無効にしておきます。
レンダリングステートの「D3DRS_FOGENABLE」(またはFogEnable)をFALSEにしてください。

(２)「フォグの基本式」と「(フォグの強さの)線形公式」をもとに、視点と頂点の距離に比例してフォグを掛けてみましょう。フォグは、「平行的なフォグ」とします。

ヒント１：バーテックスシェーダーは、以下の流れになります
１．座標変換　２．頂点色の計算　３．フォグの強さの計算　４．頂点色とフォグの合成

ヒント２：シェーダーのグローバル変数として必要なものを考え、定義します
ヒント３：線形公式では、最低限以下の情報が必要です。(頂点座標は自動的に渡されてきます)
１．フォグ色　２．フォグ開始点　３．フォグ終了点　４．視点(カメラ)座標

ヒント４：視点を平面(直線)と見立てるため、視点と頂点の距離は内積(dot関数)で求めます

ヒント５：距離を求めたら、「線形公式」より「フォグの強さ」を求めます
ヒント６：フォグの強さは、0.0f～1.0fの範囲にします
ヒント７：clamp関数を用いると、値を特定の範囲内に納めることができます

ヒント８：フォグの強さが求まったら、頂点色とフォグ色を「フォグの基本式」によって合成します

(３)「線形公式」を変形し、シェーダーの負担を減らしましょう。

線形公式を変形することにより、あらかじめプログラム側でかなりの部分を計算しておくことができるようになります(もとの公式でも"end - start"の部分はあらかじめ計算しておくことができます)。

以下のように変形します。

$$\text{フォグの強さ} = \frac{end - d}{end - start} = \frac{end}{end - start} + d * \left(\frac{-1}{end - start}\right)$$

点線部分は、あらかじめプログラム側で計算しておくことができます。シェーダーを呼び出す前にプログラム側で計算し、シェーダーのグローバル変数に設定しておけば、シェーダーで「フォグの強さ」を計算する部分が「乗算１回と足し算１回」のみになります。もともとの公式では、"end - start"をあらかじめ計算しておいたとしても、「引き算１回と除算１回」の計算が必要でした。乗算より除算の方がコストがかかるので、若干ですが負担を減らすことができます。頂点の分だけ負担が減るので、それなりの効果は期待できます(そのかわり、一目みただけではよくわからない式になってしまいます)。

(４)平行的なフォグから「距離ベースのフォグ」に変更しましょう。
ヒント：length関数(, distance関数)

(５)「指数公式」を用いたフォグに変更しましょう。
ヒント：exp関数

(６)「指数公式(２乗)」を用いたフォグに変更しましょう。
ヒント：exp関数を２回呼び出すよりは、変数に保存しておいた方が良いでしょう

応用問題　地形モデルを読み込んで視点は上から見下ろすようにし、高さフォグを掛けてみましょう。