

Tutorial 15: Vertex Fog

by Victor Saar

Content

This tutorial covers an easy way of creating fog in Direct3D. Sometimes it is also used to hide distant objects needed to achieve high framerates in realtime applications. Direct3D consists of two types of fog called vertex and pixel fog. Vertex fog is faster, as it is only computed per vertex and then interpolated over the triangle. We simply can activate vertex fog through the device's render states. There are several parameters that can be set such as color, start and end distance and the fog formula.

Normally the GPU computes for each vertex the distance from the view plane, but this can lead to visual artifacts when changing the view direction. In the first image the object is in the fog region, but after the rotation it lies outside. Some graphic cards support an additional feature called range fog. When using range fog the GPU computes the distance from the view point rather than the view plane. Therefore the object remains in the fog region no matter how the view is oriented.

capplication.cpp

All new code will be added to `InitScene()`, because the render states have to be reset in association with the lost device problem. First we create two variables defining the start and end distance of the fog area. Each object, which is farther away than the end distance is totally covered by fog.

```
float fFogStart = 2.0f, fFogEnd = 10.0f;
```

Here we come to the important part of the project. First we simply enable vertex and range fog. Then we define the fog color. The vertex mode is just a function that is used to interpolate between adjacent vertices. You can choose between linear and exponential functions. This time simple linear interpolation is used. With the last two calls we set the start and end distance for the fog. Don't worry about the strange pointer arithmetic. This is due to the fact that Direct3D internally works with `DWORD`.

```
m_pDirect3DDevice->SetRenderState(D3DRS_FOGENABLE, true);  
m_pDirect3DDevice->SetRenderState(D3DRS_RANGEFOGENABLE, true);  
m_pDirect3DDevice->SetRenderState(D3DRS_FOGCOLOR, D3DCOLOR_XRGB(128, 128, 128));  
m_pDirect3DDevice->SetRenderState(D3DRS_FOGVERTEXMODE, D3DFOG_LINEAR);  
m_pDirect3DDevice->SetRenderState(D3DRS_FOGSTART, *(DWORD*)(&fFogStart));  
m_pDirect3DDevice->SetRenderState(D3DRS_FOGEND, *(DWORD*)(&fFogEnd));
```

Furthermore we have to make sure that range-based fog is supported by the current hardware. Therefore the `RasterCaps` member of the `D3DCAPS9` structure has to be checked.

```
void CApplication::CheckDeviceCaps(void)  
{  
    m_pDirect3DDevice->GetDeviceCaps(&m_DeviceCaps);  
  
    if(!(m_DeviceCaps.RasterCaps & D3DPRASTERCAPS_FOGRANGE))  
    {  
        MessageBox(m_hWindow,  
            "range-based fog not supported!",  
            "CheckDeviceCaps()",  
            MB_OK);  
    }  
}
```

```
m_bRunningD3D = false;  
}  
} //CheckDeviceCaps
```

Direct3D's vertex fog is one of the simpler effects, but not comparable to real volumetric fog. Since vertex fog is only computed per vertex the color the back buffer is cleared with is the same as the fog color, because the fog won't be computed for the background.