

COMP3201 – Computer Graphics

Module 3: Realism and Performance

3.5 Fog

3.5.1 Introduction

The term "fog" in OpenGL is a generic term used to define a generic atmospheric type of effect. As you might expect, a white "fog" can be used to simulate real fog. Other colour fog is useful to simulate other effects, e.g., a brown colour could be used for pollution, a blue for underwater scenes.

Fog is also useful to provide depth cues and realism to a scene. Our eyes expect objects that are further away to be fuzzy. Without fog in a scene, objects at any depth are crystal clear. Fog is OpenGL's way of providing the required fuzziness.

Luckily, fog is one of the easiest effects to include in an OpenGL scene. There are three easy steps to fog.

1. Enable the fog calculation with `glEnable(GL_FOG);`
2. Specify the colour and density with calls to `glFog*()`;
3. (Optional) Give OpenGL a hint about how it should do the fog calculations.

Let us examine Step 2 further. `glFog*()` comes in two forms

```
void glFog{if}(GLenum paramname, TYPE param);  
void glFog{if}v(GLenum paramname, TYPE* params);
```

If *paramname* is one of {`GL_FOG_DENSITY`, `GL_FOG_START`, `GL_FOG_END`, `GL_FOG_MODE` } then the non-vector version of `glFog*()` is usually used. For `GL_FOG_MODE`, *param* can be one of {`GL_EXP` (the default), `GL_EXP2`, `GL_LINEAR` }.

If *paramname* is `GL_FOG_COLOR` then the vector version of `glFog*()` must be used.

Example

To set the colour of the fog red

```
GLfloat pRed[] = { 1.0, 0.0, 0.0, 1.0 };  
glFogfv( GL_FOG_COLOR, pRed );
```

3.5.2 Fog Calculation Details

The effect of fog on the colour of a fragment is determined by the following equation

$$C = f * C_i + (1-f) * C_f$$

where

- f is the fog factor due to the density (described in detail below),
- C_i is the colour of the fragment before fog effects
- C_f is the colour of the fog

The fog factor, f , encodes the density variation of the fog. Each of the different fog modes (GL_EXP, GL_EXP2, GL_LINEAR) calculates f in a different manner. These calculations are as follows

$$\begin{aligned} \text{GL_EXP:} & \quad f = \exp(-\text{density} * s) \\ \text{GL_EXP2:} & \quad f = \exp(-(\text{density} * s)^2) \\ \text{GL_LINEAR:} & \quad f = (\text{end} - s) / (\text{end} - \text{start}) \end{aligned}$$

where s is the distance from the viewpoint and the fragment centre in the eye coordinate system. Remember that the viewpoint is always at (0, 0, 0, 1) in the eye coordinate system. The values for density, start and end are set with `glFog*`(`end`, `start`, `density`);. The final value for f is always restricted to the range [0,1].

Example

To use a linearly increasing fog (assuming the colour has already been set)

```
glFogi( GL_FOG_MODE, GL_LINEAR );
glFogf( GL_FOG_DENSITY, 0.7 );
glFogf( GL_FOG_START, nearClip );
glFogf( GL_FOG_END, farClip );
```

To use an exponentially increasing fog

```
glFogi( GL_FOG_MODE, GL_EXP );
glFogf( GL_FOG_DENSITY, 0.3 );
```

Let us now examine Step 3. It is possible to give the OpenGL engine a hint on how to do the fog calculations. Since it is only a hint, a particular implementation may choose to ignore it. There are three possible fog hints. GL_NICEST asks the engine to use per pixel fog calculations and GL_FASTEST asks the engine to use per vertex fog calculations and GL_DONT_CARE (default) leaves the choice up to the engine.

Example

```
glHint( GL_FOG_HINT, GL_NICEST );
```