OpenGL|ES Tutorials (6): Fog

Fog

Funnily enough, this is used to create a fog effect. This Principle is pretty straight forward. There are several fog properties which you can set to manipulate the fog.

Fog Density (GL_FOG_DENSITY)

Thickness of the fog. The denser it is, the harder it is to see objects that are engulfed by the fog.

Fog Start (GL_FOG_START)

This is the cut off point to how near the fog is drawn. Just like the “nearz” variable that you have been using in all of the tutorials as the camera’s close up cut off point. No Fog will be render before this value.

Fog End (GL_FOG_END)

This is the cut off point to how far the fog is drawn. Just like the “farz” variable that you have been using in all of the tutorials as the camera’s distant cut off point. No Fog will be render after this value.

Fog Colour (GL_FOG_COLOR)

The Colour of the Fog.

Fog Quality (GL_FOG_HINT)

You can use the glHint command, to set the Fog Quality. There are three acceptable values: GL_DONT_CARE, GL_NICEST or GL_FASTEST. The Default is GL_DONT_CARE which basically means to make the fog as good looking as possible, if there is enough time. If you want to make your Fog look as good as possible, regardless of speed, use GL_NICEST. If you are more interested in the speed, use: GL_FATEST.

Introduction

The entire class is going to be re-written in this tutorial. However, only those lines which are new will be analysed. By now, you should be able to understand what the other lines do by yourself.

Declarations

```
#include <windows.h>
#include <GLES\gl.h>
#include <GLES\egl.h>
#include <KGSDK\framework.h>
#include <KGSDK\frameworkGL.h>

#pragma comment(lib, "Framework.lib")
#pragma comment(lib, "libGLES_CM.lib")
```

```
 9:
10:     static float rot[3];
11:     static float loc[3];
12:
13:     float LightGray[] = { 0.6f, 0.6f, 0.6f, 0.0f };    //Light Gray Colour (RGBA)
14:     float DarkGray[] = { 0.1f, 0.1f, 0.1f, 0.0f };     //Dark Gray Colour (RGBA)
15:
16:     bool ColourLight = true;         //Fog Colour Light Gray?
17:     bool EnableLighting = true;      //Lighting Enabled?
18:
19:     float xrot = 0.0f;
20:     float yrot = 0.0f;
21:
22:     float lightAmbient[] = { 0.5f, 0.5f, 0.5f, 1.0f };
23:
24:     float matAmbient[] = { 0.5f, 0.5f, 0.5f, 1.0f };
```

Analysis: The only additions here are lines 13, 14, 16, and 17. Line 13 is an array of type float, which represents a Light Gray Colour. It is in the format RGBA. Line 14, similarly is also an array of type float, but represents a Dark Gray Colour. Line 16 declares a Boolean, which is initialised to true. This will keep record of whether we are using a Dark Coloured fog (false), or a Light Coloured fog (true). Line 17 declares another Boolean to keep track of whether or not Lighting is enabled (true).

Init

```
 1:    void Init()
 2:    {
 3:          glLoadIdentity();
 4:
 5:          GLfixed mat[4][4];
 6:
 7:          float nearz, farz;
 8:
 9:          farz  = 300.0f;
10:          nearz = 0.01f;
11:
12:          memset(mat, 0, sizeof(mat));
13:          mat[0][0] = (int) (65536.0f * 2.4f);
14:          mat[1][1] = (int) (65536.0f * 3.2f);
15:          mat[2][2] = (int) (65536.0f * (farz / (farz - nearz)));
16:          mat[2][3] = (int) (65536.0f * 1.0f);
17:          mat[3][2] = (int) (65536.0f * ((-farz * nearz) / (farz - nearz)));

18:
19:          glMatrixMode(GL_PROJECTION);
20:          glLoadMatrixx(&mat[0][0]);
21:
22:          rot[0]  = 0;
23:          rot[1]  = 0;
24:          rot[2]  = 0;
25:          loc[0]  = 0;
26:          loc[1]  = 0;
27:          loc[2]  = 3;
28:
29:          glFogf(GL_FOG_MODE, GL_LINEAR);     //Set Fog Mode
30:         glFogfv(GL_FOG_COLOR, LightGray);  //Set Fog Colour to Light Gray
31:         glFogf(GL_FOG_DENSITY, 0.4f);      //Set Fog Density
32:         glHint(GL_FOG_HINT, GL_NICEST);    //Set Fog Quality
33:         glFogf(GL_FOG_START, 1.0f);        //Set Fog "nearz" value
34:         glFogf(GL_FOG_END, 5.0f);          //Set Fog "farz" value
35:         glEnable(GL_FOG);                  //Enable Fog Capability
36:
37:         glEnable(GL_LIGHTING);             //Enable Lighting
38:         glEnable(GL_LIGHT0);               //Enable Light 0
39:         glEnable(GL_TEXTURE_2D);           //Enable Textures
40:
41:         glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, matAmbient);
42:
43:         glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
44:
45:         glShadeModel(GL_SMOOTH);
46:   }
```

Analysis: Lines 29 – 35 have been added. In this section of code we initialise the Fog. There are certain properties for it which we can set. To do this we use the glFog or glFogfv function. The first parameter in both of these functions is what property you wish to set, and the second is the value of the property you wish to set it to. Line 29 sets "Fog Mode" to GL_LINEAR. Fog Modes were discussed at the beginning of this tutorial, along with all the other properties and values. Next we set the Fog Colour to Light Gray. Note that this is the only time we use glFogfv. Line 31 sets the Fog Density to 0.4f and line 32 sets the fog quality. Lines 33 and 34 set the Fog close-up cut off point, and the distant cut-off point, respectively. Finally, as with all OpenGL|ES capabilities, we enable the Fog capability on line 35, by using the glEnable function.

Function: DrawCube()

```
 1:      void DrawCube()
 2:      {
 3:         GLfloat cube[] =
 4:         {
 5:         // FRONT
 6:         -0.5f, -0.5f,  0.5f,
 7:          0.5f, -0.5f,  0.5f,
 8:         -0.5f,  0.5f,  0.5f,
 9:          0.5f,  0.5f,  0.5f,
10:         // BACK
11:         -0.5f, -0.5f, -0.5f,
12:         -0.5f,  0.5f, -0.5f,
13:          0.5f, -0.5f, -0.5f,
14:          0.5f,  0.5f, -0.5f,
15:         // LEFT
16:         -0.5f, -0.5f,  0.5f,
17:         -0.5f,  0.5f,  0.5f,
18:         -0.5f, -0.5f, -0.5f,
19:         -0.5f,  0.5f, -0.5f,
20:         // RIGHT
21:          0.5f, -0.5f, -0.5f,
22:          0.5f,  0.5f, -0.5f,
23:          0.5f, -0.5f,  0.5f,
24:          0.5f,  0.5f,  0.5f,
25:         // TOP
26:         -0.5f,  0.5f,  0.5f,
27:          0.5f,  0.5f,  0.5f,
28:         -0.5f,  0.5f, -0.5f,
         0.5f,  0.5f, -0.5f,
        // BOTTOM
        -0.5f, -0.5f,  0.5f,
        -0.5f, -0.5f, -0.5f,
         0.5f, -0.5f,  0.5f,
         0.5f, -0.5f, -0.5f,
        };

        GLfloat texCoords[] = {
        // FRONT
         0.0f, 0.0f,
         1.0f, 0.0f,
         0.0f, 1.0f,
         1.0f, 1.0f,
        // BACK
         1.0f, 0.0f,
         1.0f, 1.0f,
         0.0f, 0.0f,
         0.0f, 1.0f,
        // LEFT
         1.0f, 0.0f,
         1.0f, 1.0f,
         0.0f, 0.0f,
         0.0f, 1.0f,
        // RIGHT
         1.0f, 0.0f,
         1.0f, 1.0f,
         0.0f, 0.0f,
         0.0f, 1.0f,
        // TOP
         0.0f, 0.0f,
         1.0f, 0.0f,
         0.0f, 1.0f,
         1.0f, 1.0f,
        // BOTTOM
         1.0f, 0.0f,
         1.0f, 1.0f,
         0.0f, 0.0f,
         0.0f, 1.0f
        };

        glRotatef(xrot, 1.0f, 0.0f, 0.0f);
        glRotatef(yrot, 0.0f, 1.0f, 0.0f);

        glVertexPointer(3, GL_FLOAT, 0, cube);
        glTexCoordPointer(2, GL_FLOAT, 0, texCoords);
        glEnableClientState(GL_VERTEX_ARRAY);
        glEnableClientState(GL_TEXTURE_COORD_ARRAY);

       // FRONT AND BACK
       glColor4f(1.0f, 0.0f, 0.0f, 1.0f);             //Color: RED
       glNormal3f(0.0f, 0.0f, 1.0f);
       glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
       glNormal3f(0.0f, 0.0f,-1.0f);
       glDrawArrays(GL_TRIANGLE_STRIP, 4, 4);

       // LEFT AND RIGHT
       glColor4f(0.0f, 1.0f, 0.0f, 1.0f);             //Color: GREEN
       glNormal3f(-1.0f, 0.0f, 0.0f);
       glDrawArrays(GL_TRIANGLE_STRIP, 8, 4);
       glNormal3f(1.0f, 0.0f, 1.0f);
       glDrawArrays(GL_TRIANGLE_STRIP, 12, 4);

      // TOP AND BOTTOM
        glColor4f(0.0f, 0.0f, 1.0f, 1.0f);            //Color: BLUE
        glNormal3f(0.0f, 1.0f, 0.0f);
        glDrawArrays(GL_TRIANGLE_STRIP, 16, 4);
        glNormal3f(0.0f,-1.0f, 1.0f);
        glDrawArrays(GL_TRIANGLE_STRIP, 20, 4);
    }
```

Analysis: This function has remained very much unchanged. You should be able to interpret this section by yourself. Only one thing to point out here is: Why have we enabled textures, when we're not going to map any image? The reason is that when I tested Fog without textures enabled on my Gizmondo, it failed to work. I do not know exactly the reason for this. In addition, when I mapped and image onto the sides of the cube, the fog had disappeared. This means that you are currently unable to use Fog and textures together. If you know the reason to this, or have successfully managed to use textures and Fog together on your Gizmondo, please contact me (King@GizmondoForums.com).

Render

```
void Render()
{
  glClearColor(0.6f, 0.6f, 0.6f, 1.0f);   //Background: Light Gray
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  DrawCube();

  glFinish();
  eglSwapBuffers(FrameworkGL_GetDisplay(), FrameworkGL_GetSurface());
}
```

Analysis: Again, little modification has been made to this function. The only difference is this tutorial is that the background colour is set to the same colour as the Light Gray colour we defined earlier in the declarations section. This is essential for Fog as you will see later on.

Update

```
void Update()
{
  GLfixed mat[4][4];

  memset(mat, 0, sizeof(mat));
  mat[0][0] = (int) (65536.0f *  cos(rot[0]));
  mat[0][2] = (int) (65536.0f *  sin(rot[0]));
  mat[1][1] = (int) (65536.0f *  cos(rot[1]));
  mat[1][2] = (int) (65536.0f *  sin(rot[1]));
  mat[2][0] = (int) (65536.0f * -sin(rot[0]) * cos(rot[1]));
  mat[2][2] = (int) (65536.0f *  cos(rot[0]) * cos(rot[1]));
  mat[3][2] = (int) (65536.0f * loc[2]);
  mat[3][3] = 65536;

  glMatrixMode(GL_MODELVIEW);
  glLoadMatrixx(&mat[0][0]);

  if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_LEFT_SHOULDER ))
     loc[2] += 0.3f;
20:    // If the Left Shoulder Button has been pressed: Zoom out
21:    if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_RIGHT_SHOULDER))
22:        loc[2] -= 0.3f;
23:    // If the Right Shoulder Button has been pressed: Zoom In
24:    if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_DPAD_DOWN     ))
25:        xrot -= 1.0f;
26:    // If the Down button (DPAD) has been pressed: Rotate the cube downards
27:    if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_DPAD_UP         ))
28:        xrot += 1.0f;
29:    // If the Up button (DPAD) has been pressed: Rotate the cube upwards
30:    if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_DPAD_LEFT     ))
31:        yrot -= 1.0f;
32:    // If the Left button (DPAD) has been pressed: Rotate the cube towards
the left
33:    if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_DPAD_RIGHT  ))
34:        yrot += 1.0f;
35:    // If the Right button (DPAD) has been pressed: Rotate the cube towards
the Right
36:    if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_FORWARD     ))
37:    {
38:        EnableLighting =! EnableLighting;
39:        (EnableLighting ? glEnable(GL_LIGHTING) : glDisable(GL_LIGHTING));
40:    }
41:    // If the Forward has been pressed: Toggle Lighting
42:    if (Framework_IsButtonPressed(FRAMEWORK_BUTTON_REWIND      ))
43:    {
44:        ColourLight =! ColourLight;
45:        (ColourLight ? glFogfv(GL_FOG_COLOR, LightGray) :
glFogfv(GL_FOG_COLOR, DarkGray));
46:    }
47:    // If the Rewind has been pressed: Toggle Colour
48: }
```

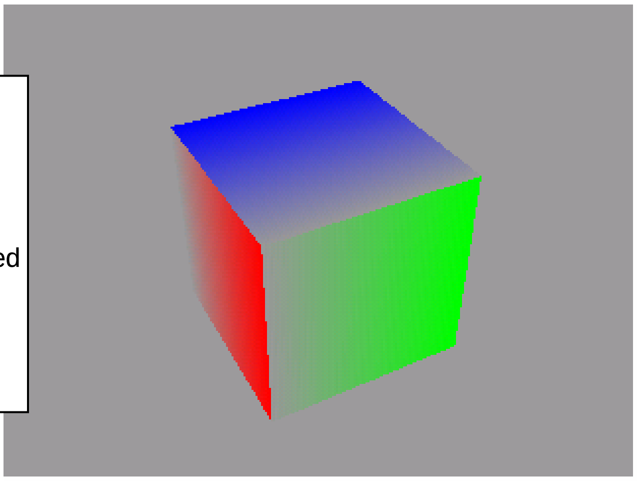
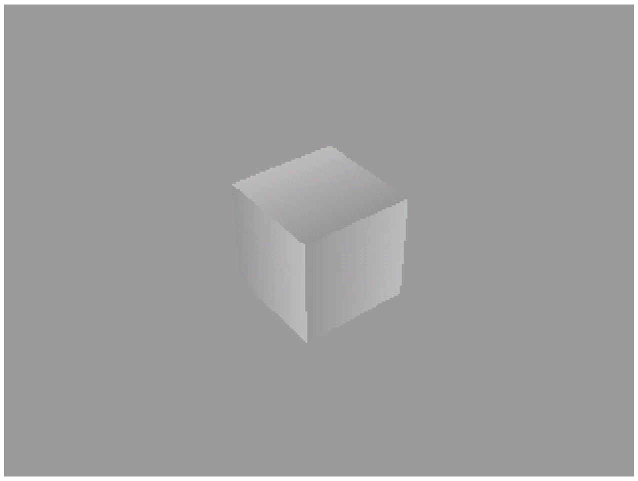
Analysis: We've added code here to see whether the REWIND button, and the FORWARD button has been pressed. Lines 36 – 40 Toggle Lighting, when the FORWARD key has been pressed, and lines 42 – 46 Toggle the Fog Colour, when the REWIND key has been pressed.

Output

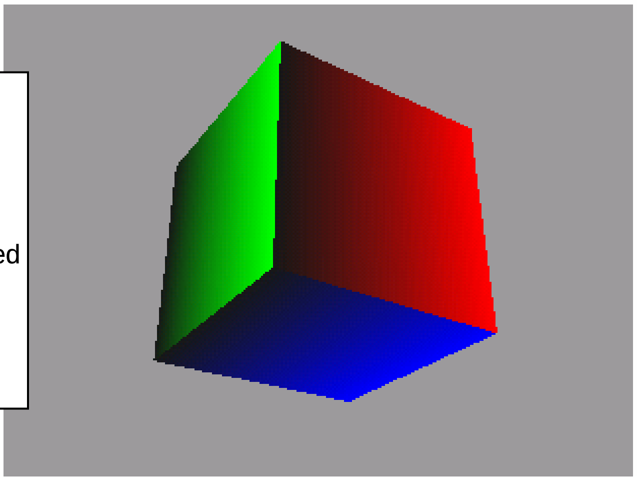
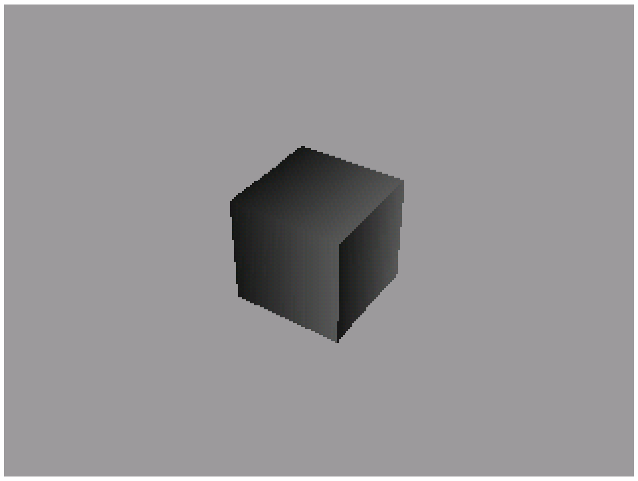
Lighting Disabled:

Lighting Disabled:

Light Gray Coloured Fog:

Dark Gray Coloured Fog:

As you can see, for the Fog “effect” to work, the Fog Colour needs to be the same as the background (compare Top Row with Bottom Row). In addition it is advisable lighting is not used, (depending on the effect you’re trying to create).

- End of OpenGL|ES Tutorials: Fog-