OpenGL|ES Tutorials (5): Texture Filters

Texture Filters

Texture Filters simply allow you to manipulate your texture. Some of the aspects of the textures which can be modified are discussed below.

Repeating and Clamping

When you created a texture in the last tutorial you simply mapped the picture straight onto the faces of the cube. However, it may desirable that you wish to stretch an image or repeat it over and over again. This is what repeating and clamping is.

MipMap Generation

MipMap is an important filter in OpenGL|ES. What mipmap does is generate different size textures based on the viewer's distance from the object. For example, as you zoom out and the cube gets smaller, OpenGL generates different size textures (64x64 → 32x32 → 16x16, etc). This increases the "smoothness" of your game, but at the same time reduces the overall performance.

Introduction

This tutorial uses the code from the first part of this tutorial. We will be making little modifications this time.

Function: LoadTextures

```
1:      bool LoadTextures()
2:      {
3:        BITMAPINFOHEADER info;              //Structure to hold bitmap data
4:        unsigned char *bitmap = NULL;       //Pointer to point to image data
5:
6:        bitmap = LoadBitmap("King.bmp", &info);    //Load Bitmap
7:
8:        if (!bitmap)
9:        return false;
10:
11:       glGenTextures(4, texture);          //Generate Texture Identifiers
12:
13:       //Tex-1
14:       glBindTexture(GL_TEXTURE_2D, texture[0]);  //Select texture[0]
15:
16:       glTexImage2D(
17:         GL_TEXTURE_2D, 0, GL_RGB, info.biWidth   //Set Texture Properties
18:         info.biHeight, 0, GL_RGB, GL_UNSIGNED_BYTE,
19:         bitmap);
20:
21:       //Set texture filter to GL_NEAREST when texture is small
22:       glTexParameterf(GL_TEXTURE_2D,
```

```
23:           GL_TEXTURE_MIN_FILTER, GL_NEAREST);
24:         //Set texture filter to GL_NEAREST when texture is big
25:         glTexParameterf(GL_TEXTURE_2D,
26:           GL_TEXTURE_MAG_FILTER, GL_NEAREST);
27:
28:         //Tex-2
29:         glBindTexture(GL_TEXTURE_2D, texture[1]);  //Select texture[1]
30:
31:         glTexImage2D(                              //Set Texture Properties
32:           GL_TEXTURE_2D, 0, GL_RGB, info.biWidth,
33:           info.biHeight, 0, GL_RGB, GL_UNSIGNED_BYTE,
34:           bitmap);
35:
36:         //Set texture filter to GL_LINEAR when texture is small
37:         glTexParameterf(GL_TEXTURE_2D,
38:           GL_TEXTURE_MIN_FILTER, GL_LINEAR);
39:         //Set texture filter to GL_LINEAR when texture is big
40:         glTexParameterf(GL_TEXTURE_2D,
41:           GL_TEXTURE_MAG_FILTER, GL_LINEAR);
42:
43:         // Tex-3
44:         glBindTexture(GL_TEXTURE_2D, texture[2]);  //Select texture[2]
45:
46:         glTexImage2D(                              //Set Texture Properties
47:           GL_TEXTURE_2D, 0, GL_RGB, info.biWidth,
48:           info.biHeight, 0, GL_RGB, GL_UNSIGNED_BYTE,
49:           bitmap);
50:
51:         //Set texture filter to GL_LINEAR when texture is small
52:         glTexParameterf(GL_TEXTURE_2D,
53:           GL_TEXTURE_MIN_FILTER, GL_LINEAR);
54:         //Set texture filter to GL_LINEAR when texture is big
55:         glTexParameterf(GL_TEXTURE_2D,
56:           GL_TEXTURE_MAG_FILTER, GL_LINEAR);
57:         //Set texture to repeat itself horizontally
58:         glTexParameterf(GL_TEXTURE_2D,
59:           GL_TEXTURE_WRAP_S, GL_REPEAT);
60:         //Set texture to clamp to edges vertically
61:         glTexParameterf(GL_TEXTURE_2D,
62:           GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
63:
64:         // Tex-4
65:         glBindTexture(GL_TEXTURE_2D, texture[3]);  //Select texture[3]
66:
67:         glTexImage2D(                              //Set Texture Properties
68:           GL_TEXTURE_2D, 0, GL_RGB, info.biWidth,
69:           info.biHeight, 0, GL_RGB, GL_UNSIGNED_BYTE,
70:           bitmap);
71:         //Set texture filter to GL_LINEAR when texture is small
72:         glTexParameterf(GL_TEXTURE_2D,
73:           GL_TEXTURE_MIN_FILTER, GL_LINEAR);
74:         //Set texture filter to GL_LINEAR when texture is big
75:         glTexParameterf(GL_TEXTURE_2D,
76:           GL_TEXTURE_MAG_FILTER, GL_LINEAR);
77:         glTexParameterf(GL_TEXTURE_2D,       //Enable MipMap Generation
78:           GL_GENERATE_MIPMAP, GL_TRUE);
79:
80:         delete[] bitmap;                     //Clean up
81:
82:         return true;
83:       }
```

Analysis: This time instead of creating one texture, we create four different textures with different properties. This Analysis will only discuss the textures part of this function (Lines 13 – 78).

The first texture: Lines 13 – 26, creates a simple texture. On line 14 texture[0] is selected, and then the glTexImage2D function is used (See previous tutorial for more info on this function). We then utilise the function glTexParameterf to set the texture's filter properties. The first parameter will always be the flag: GL_TEXTURE_2D. The second parameter will specify if we want to set properties for the texture if it were up close (GL_TEXTURE_MAG_FILTER) or far away (GL_TEXTURE_MIN_FILTER). The third and final parameter will set the actual value of the property. This can be one of the following flags: GL_LINEAR or GL_NEAREST. If you use GL_LINEAR it will look better, but might slow your program down, whereas if you use GL_NEAREST it won't look as good, but your program will run faster. There are some other flags which can be used, and are discussed below.

Texture 2 (Lines 28 – 41) is exactly the same as texture one but instead of using GL_NEAREST we use GL_LINEAR. As explained before, GL_LINEAR looks better than GL_NEAREST.

Texture 3 (Lines 51 – 62) is also similar to texture 2: GL_LINEAR is used. However, two new filter properties are set as well: GL_TEXTURE_WRAP_S and GL_TEXTURE_WRAP_T. This specifies how the texture is mapped onto the surface horizontally and vertically respectively. For the GL_TEXTURE_WRAP_S property we set it too repeat itself. And for the GL_TEXTURE_WRAP_T property we set it to clamp to the edges.

Texture 4 (Lines 64 – 78) is also similar to texture 2: GL_LINEAR is used. This texture is very important though: It demonstrates mipmaps.

Render

We have changed our Render function so that we now map each texture we have created on to each side of our cube. This way you can see the differences between the textures.

```
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,
29:        0.5f,  0.5f, -0.5f,
30:       // BOTTOM
31:       -0.5f, -0.5f,  0.5f,
32:       -0.5f, -0.5f, -0.5f,
33:        0.5f, -0.5f,  0.5f,
34:        0.5f, -0.5f, -0.5f,
35:       };
36:
37:       GLfloat texCoords[] = {
38:       // FRONT
39:        0.0f, 0.0f,
40:        1.0f, 0.0f,
41:        0.0f, 1.0f,
42:        1.0f, 1.0f,
43:       // BACK
44:        1.0f, 0.0f,
45:        1.0f, 1.0f,
46:        0.0f, 0.0f,
47:        0.0f, 1.0f,
48:       // LEFT
49:        2.0f, 0.0f,
50:        2.0f, 2.0f,
51:        0.0f, 0.0f,
52:        0.0f, 2.0f,
53:       // RIGHT
54:        3.0f, 0.0f,
55:        3.0f, 3.0f,
56:        0.0f, 0.0f,
57:        0.0f, 3.0f,
58:       // TOP
59:        0.0f, 0.0f,
60:        1.0f, 0.0f,
61:        0.0f, 1.0f,
62:        1.0f, 1.0f,
63:       // BOTTOM
64:        1.0f, 0.0f,
65:        1.0f, 1.0f,
66:        0.0f, 0.0f,
67:        0.0f, 1.0f
68:       };
69:       glRotatef(xrot, 1.0f, .0f, 0.0f);
70:       glRotatef(yrot, 0.0f, 1.0f, 0.0f);
71:
72:       glVertexPointer(3, GL_FLOAT, 0, cube);
73:       glTexCoordPointer(2, GL_FLOAT, 0, texCoords);
74:       glEnableClientState(GL_VERTEX_ARRAY);
75:       glEnableClientState(GL_TEXTURE_COORD_ARRAY);
76:
77:       // FRONT AND BACK
78:       glBindTexture(GL_TEXTURE_2D, texture[0]);
79:       glColor4f(1.0f, 0.0f, 0.0f, 1.0f);     //Color: RED
```

```
80:       glNormal3f(0.0f, 0.0f, 1.0f);
81:       glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
82:       glNormal3f(0.0f, 0.0f,-1.0f);
83:       glDrawArrays(GL_TRIANGLE_STRIP, 4, 4);
84:
85:       // LEFT AND RIGHT
86:       glBindTexture(GL_TEXTURE_2D, texture[1]);
87:       glColor4f(0.0f, 1.0f, 0.0f, 1.0f);    //Color: GREEN
88:       glNormal3f(-1.0f, 0.0f, 0.0f);
89:       glDrawArrays(GL_TRIANGLE_STRIP, 8, 4);
90:       glNormal3f(1.0f, 0.0f, 1.0f);
91:       glDrawArrays(GL_TRIANGLE_STRIP, 12, 4);
92:
93:       // TOP AND BOTTOM
94:       glColor4f(0.0f, 0.0f, 1.0f, 1.0f);          //Color: BLUE

95:       glBindTexture(GL_TEXTURE_2D, texture[2]);
96:       glNormal3f(0.0f, 1.0f, 0.0f);
97:       glDrawArrays(GL_TRIANGLE_STRIP, 16, 4);
98:       glBindTexture(GL_TEXTURE_2D, texture[3]);
99:       glNormal3f(0.0f,-1.0f, 1.0f);
100:      glDrawArrays(GL_TRIANGLE_STRIP, 20, 4);
101:  }
```

Analysis: Our texCoords array has changed slightly here. We have increased some values. The Left side of the cube now has double the image, and the right side will have 3 times the image. The only other modification to this function is that before we draw each side of the face, we select a different texture each time by using the glBindTexture. (With exception to the FRONT and BACK faces, which both have the same texture.

Output

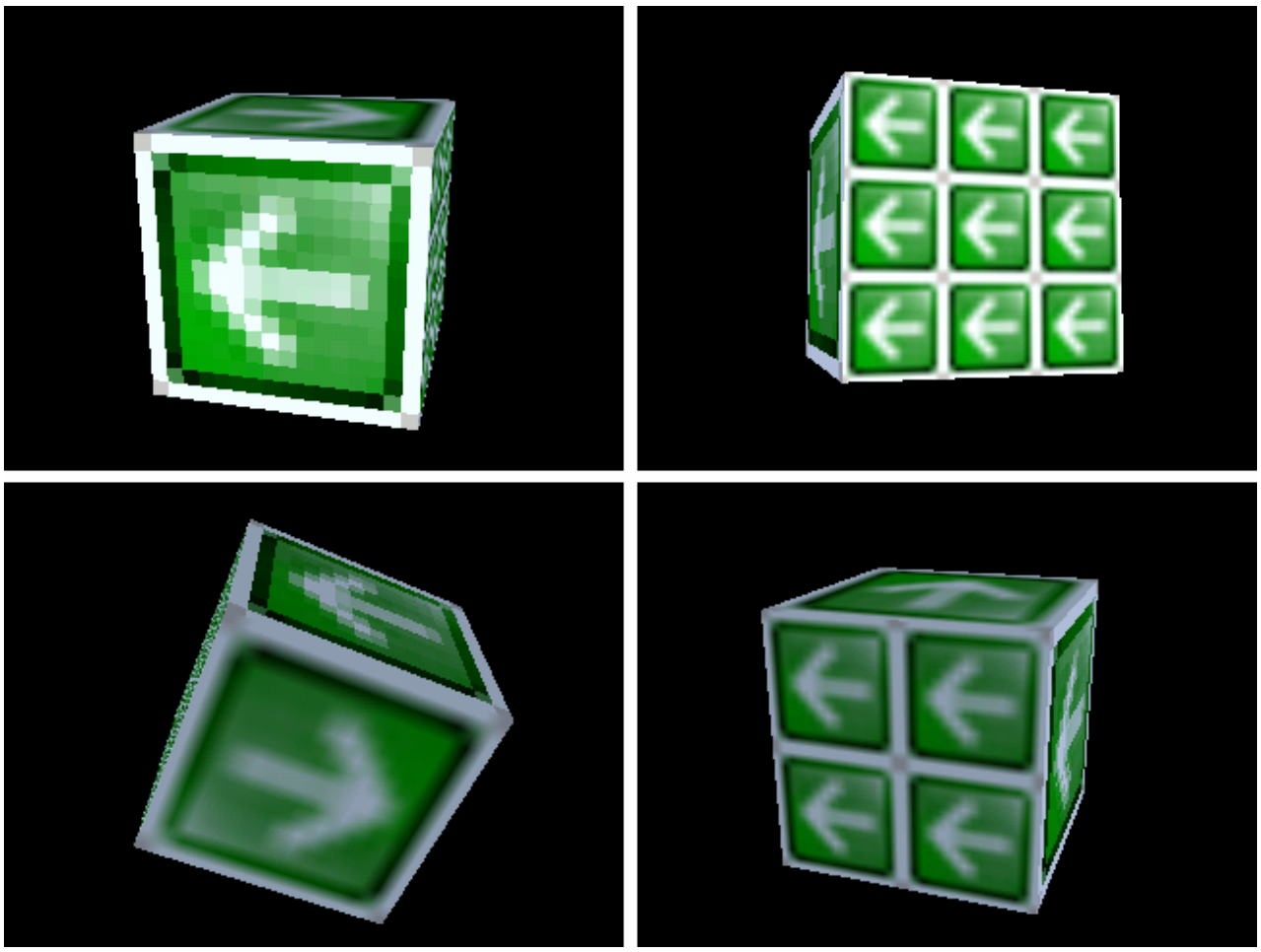

Figure 5.1 Tutorial 5 Output

- End of OpenGL|ES Tutorials: Textures Filters-