Texturing

纹理的作用

在渲染管线中，到FragmentShader以前进行的都是几何概念上的事物对象，把一个几何对象绘制成为贴近真实表现的图像，就需要用到纹理；纹理里面存储了需要绘制的图像的外表细节信息；在Fragment Shader中，把纹理绘制到相应的几何物体上，就得到了比较真实的图像。

纹理的种类

- 2d Texture: 最常用的纹理

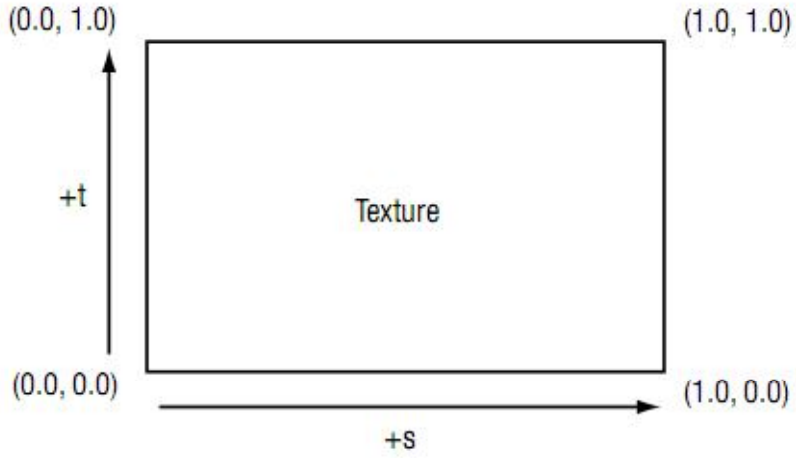


Figure 9-1 2D Texture Coordinates

- cubemap texture: 常用来表示世界背景的纹理，可以认为是放在上下左右前后六个方向无穷远处的六个2d Texture

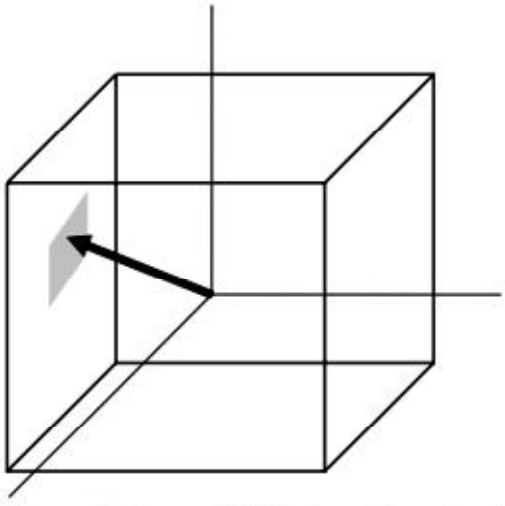


Figure 9-2 3D Texture Coordinate for Cubemap

- 3d Texture：GLES2还不支持

如何使用纹理对象

1. 生成纹理ID

```
void glGenTextures(    GLsizei n, GLuint * textures);
void glBindTexture(    GLenum target,              // Must be either GL_TEXTURE_2D o
     GLuint texture);
```

2. 载入纹理数据

```
void glPixelStorei(GL_UNPACK_ALIGNMENT, [1, 2, 4, 8]); // 设置要传入纹理数据的对齐格式

void glTexImage2D(    GLenum target, // Must be GL_TEXTURE_2D, GL_TEXTURE_CUBE_MAP_{
     GLint level,  // Level 0 is the base image level. Level n is the nth mipmap redu
     GLint internalformat, // Must be one of the following symbolic constants: GL_ALP
     GLsizei width,
     GLsizei height,
     GLint border, // Must be 0.
     GLenum format,  // Specifies the format of the texel data. Must match internalfo
     GLenum type, // Specifies the data type of the texel data. GL_UNSIGNED_BYTE, GL_
     const GLvoid * data);

// 载入压缩纹理,特定的芯片只支持特定的纹理压缩格式
void glCompressedTexImage2D(    GLenum target, // 同上.
     GLint level, // 同上.
     GLenum internalformat, // Specifies the format of the compressed image data stor
     GLsizei width,
     GLsizei height,
     GLint border, // 同上.
     GLsizei imageSize, // Specifies the number of unsigned bytes of image data start
     const GLvoid * data);
```

3. 设置纹理参数, 决定纹理数据如何被使用

```
void glTexParameterfv(    GLenum target,
     GLenum pname,
     const GLfloat * params);

void glTexParameteriv(    GLenum target,
     GLenum pname,
     const GLint * params);
target
Specifies the target texture of the active texture unit, which must be either GL_TEXT

pname
Specifies the symbolic name of a texture parameter. pname can be one of the following

params
Specifies a pointer to an array where the value of pname is stored.
```

- FILTER参数的取值: GL_NEAREST, GL_LINEAR or GL_{NEAREST, LINEAR}_MIPMAP_{NEAREST, LINEAR},

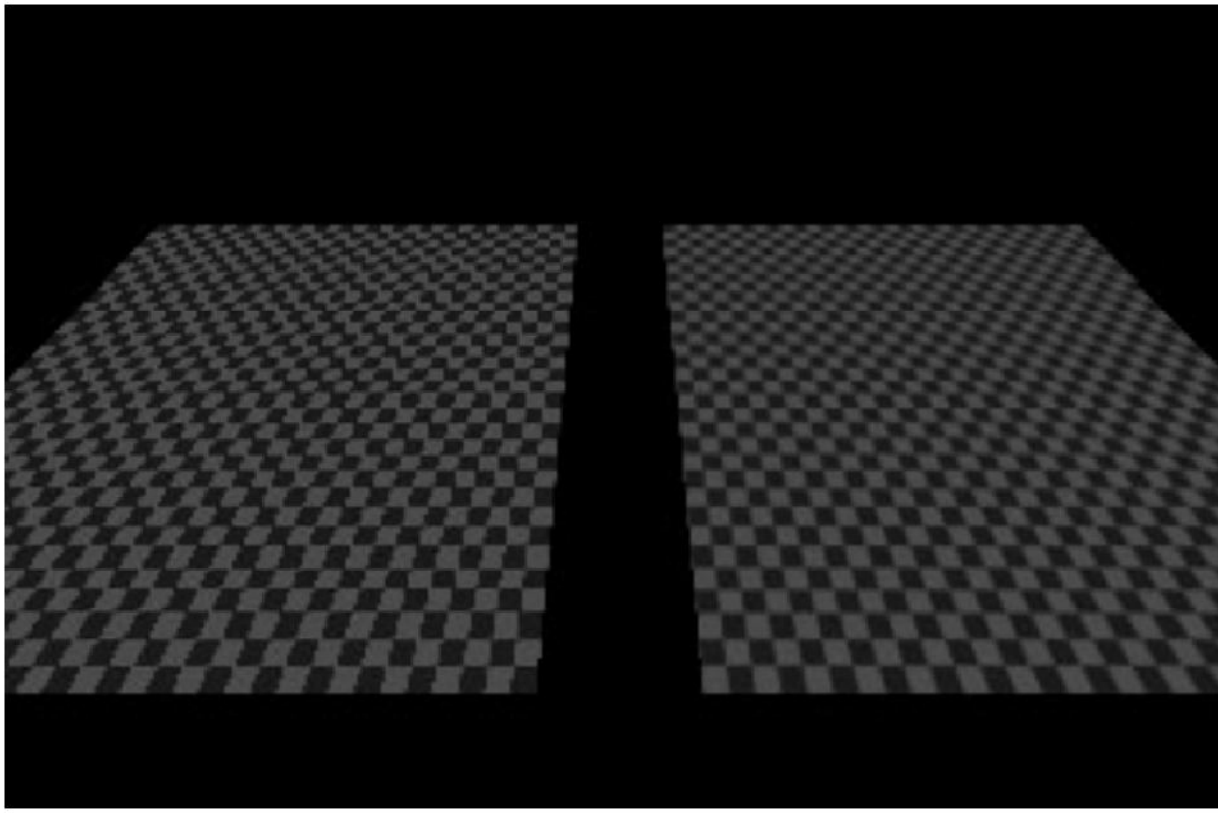
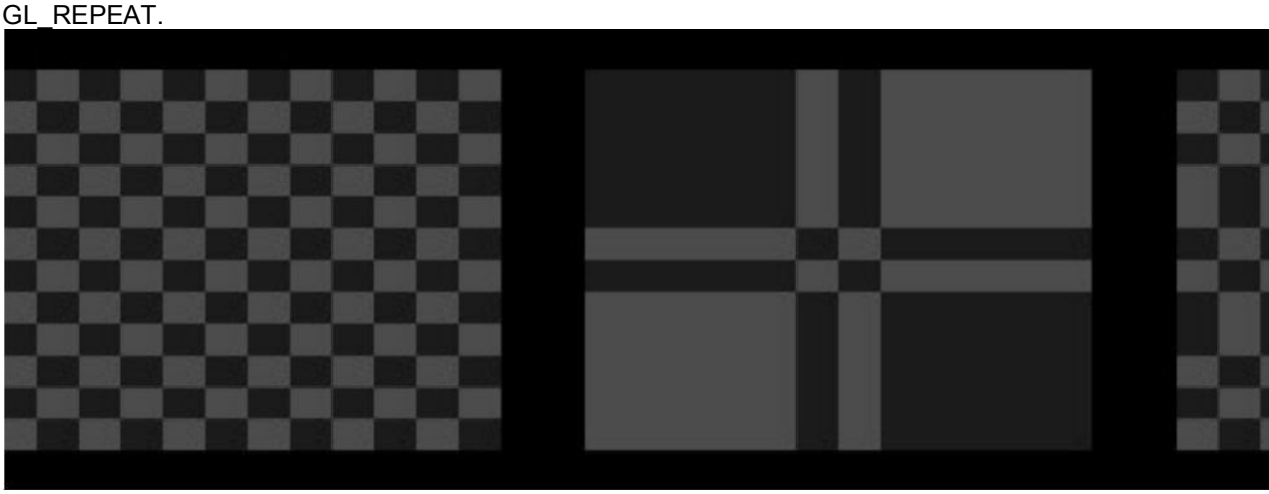


Figure 9-3 MipMap2D: Nearest Versus Trilinear Filt

- WRAP参数的取值: GL_CLAMP_TO_EDGE, GL_MIRRORED_REPEAT, or GL_REPEAT.



4. 在Fragment Shader中获取纹理的颜色填充到像素点上

```
"uniform    sampler2D       samTexture2d;"
"void main (void)"
"{"
"    gl_FragColor = texture2D(samTexture2d, gl_PointCoord);"
"}";
```

5. 纹理数据的获取

```
void glPixelStorei(GL_PACK_ALIGNMENT, [1, 2, 4, 8]); // 设置数据传出后的对齐格式

void glReadPixels(    GLint x,
     GLint y,
     GLsizei width,
     GLsizei height,
     GLenum format, // Specifies the format of the pixel data. GL_ALPHA, GL_RGB, and
     GLenum type, // Specifies the data type of the pixel data. Must be one of GL_UNS
     GLvoid * data);
```

6. 纹理数据的更新

```
// 参数同glTexImage2D
void glTexSubImage2D(    GLenum target,
     GLint level,
     GLint xoffset,
     GLint yoffset,
     GLsizei width,
     GLsizei height,
     GLenum format,
     GLenum type,
     const GLvoid * data);
```

7. 删除纹理

```
void glDeleteTextures(    GLsizei n,
     const GLuint * textures);
```

多重纹理

多重纹理是OpenGL ES 2.0 新增的特性，以前在一次绘制中，只能指定一个纹理，如果需要绘制多个纹理，就需要调用多次绘制函数。
OpenGL ES 2 最少支持同时使用8个纹理，但默认只启用GL_TEXTURE0，行为和只能使用一个纹理的情况一样，不需要特殊处理，如果要使用多重纹理，需要如下显示启用：

```
    glActiveTexture(GL_TEXTURE0);
    glUniform1i(samplerName0, 0);
    glBindTexture(GL_TEXTURE_2D, textureName0);

    glActiveTexture(GL_TEXTURE1);
    glUniform1i(SamplerName1, 1);
    glBindTexture(GL_TEXTURE_2D, textureName1);
    ......
```

- 在Fragment Shader中，纹理是一种特殊的Uniform，可以显示的指定Sampler和哪个Texture对象绑定，启用需要的Texture对象后，可以通过glBindTexture把当前的Texture对象和某个纹理绑定。
- 正确的使用多重纹理，可以提高绘制效率。
- 示例代码演示了一个栅格动画效果，每一帧使用两个纹理，但只需要绘制一次。

Mipmap

一个纹理可以有多层数据，一般一帧的图像存在level0，可以给此纹理设置第二层图像，保存在Level1，但数据的长宽是Level0的一半，还可以设置Level2的数据，长宽是Level1的一半，以此类推，只到最后一层数据的长宽其中有一个为1。

- 比如有一个64x64的纹理数据，Level0的数据为64x64，Level1可以存32x32的数据，Level2可以存16x16的数据，Level3可以存8x8的数据，Level4可以存4x4的数据，Level5可以存2x2的数据，Level6可以存1x1的数据。
- 一个纹理多层的数据可以指定硬件自动生成，也可以由代码指定数据。
- 多层纹理之间并不需要一直填充到1x1的层级，可以只有前几层。
- 一个包含多层数据的纹理称为Mipmap。

Mipmap的作用是当绘制的图像小于纹理的大小时，可以直接读取较小的Level层的数据，可以减少纹理数据对内存带宽的占用；此外，还可以使显示效果更真实。