
福州大学
FUZHOU UNIVERSITY

# 第 7 章 第二节

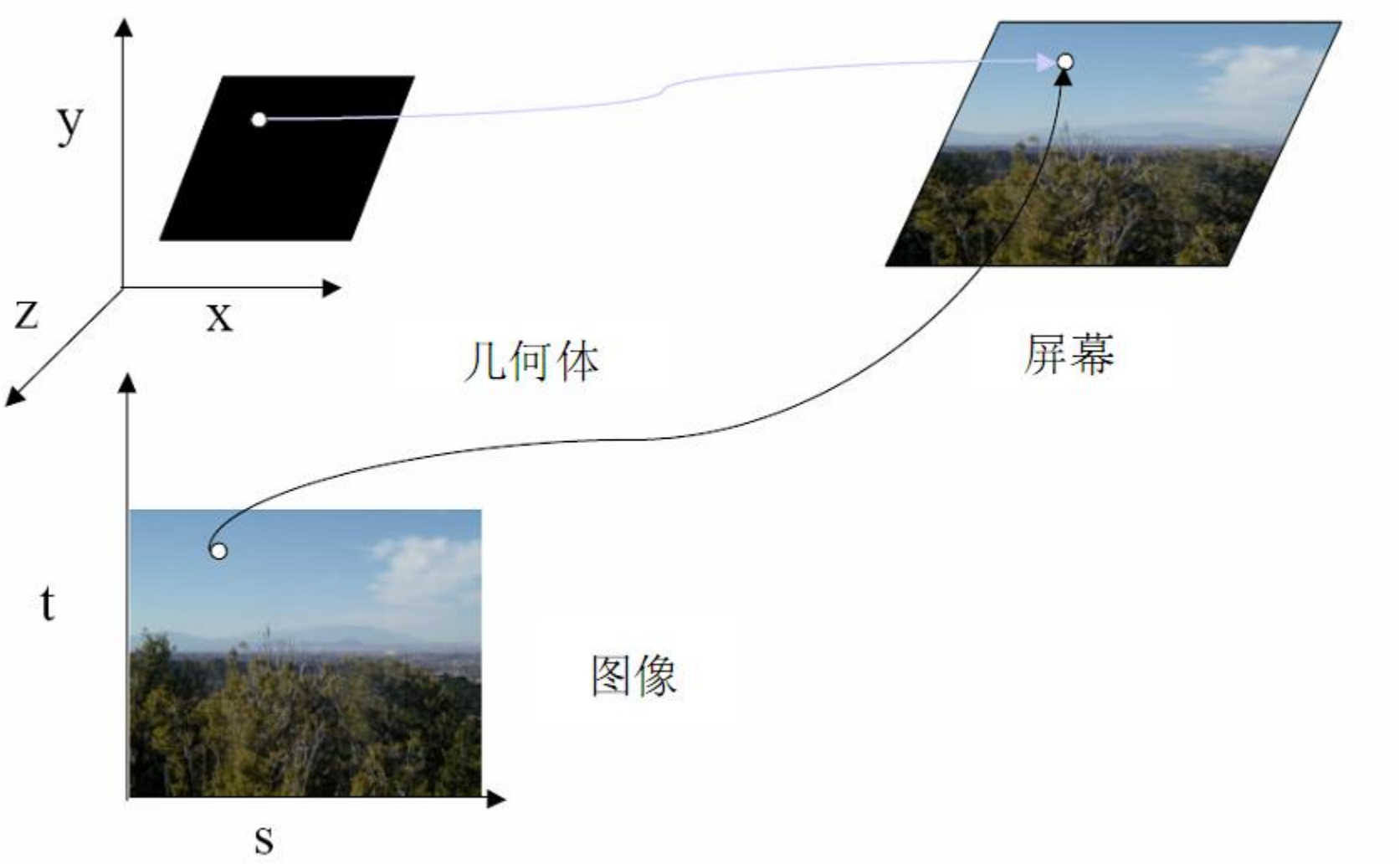
# WebGL中的纹理映射

# WebGL中的纹理映射



- 纹理映射是在光栅化后的片元处理单元汇合，纹理映射作为片元处理的一部分执行。
- 纹理映射与法向量和颜色的处理方式类似。作为WebGL的状态与顶点相关联，在多边形的表面上对顶点的纹理坐标进行插值计算，从而得到多边形表面上其他位置的纹理坐标。
- WebGL只支持二维纹理映射

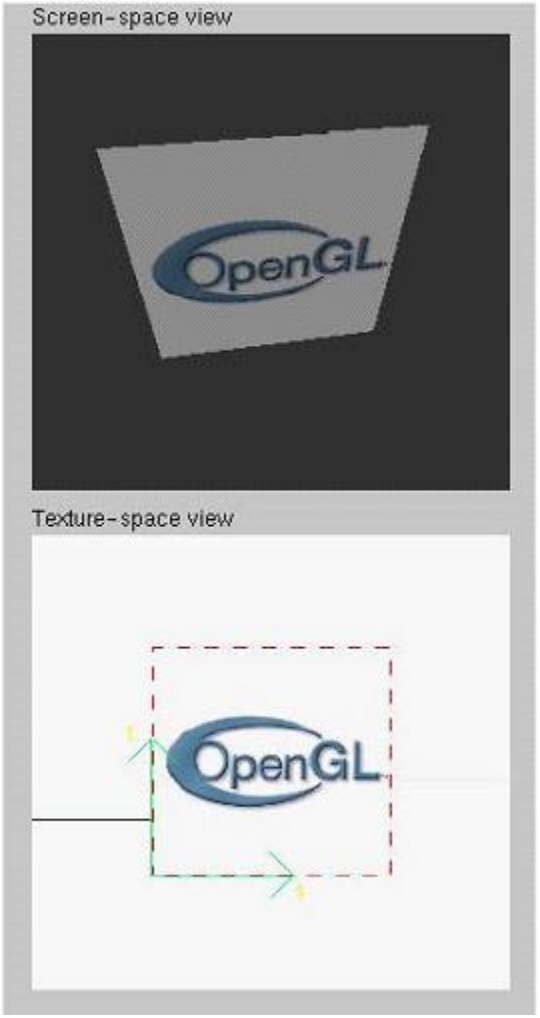
# 纹理映射



# 纹理示例



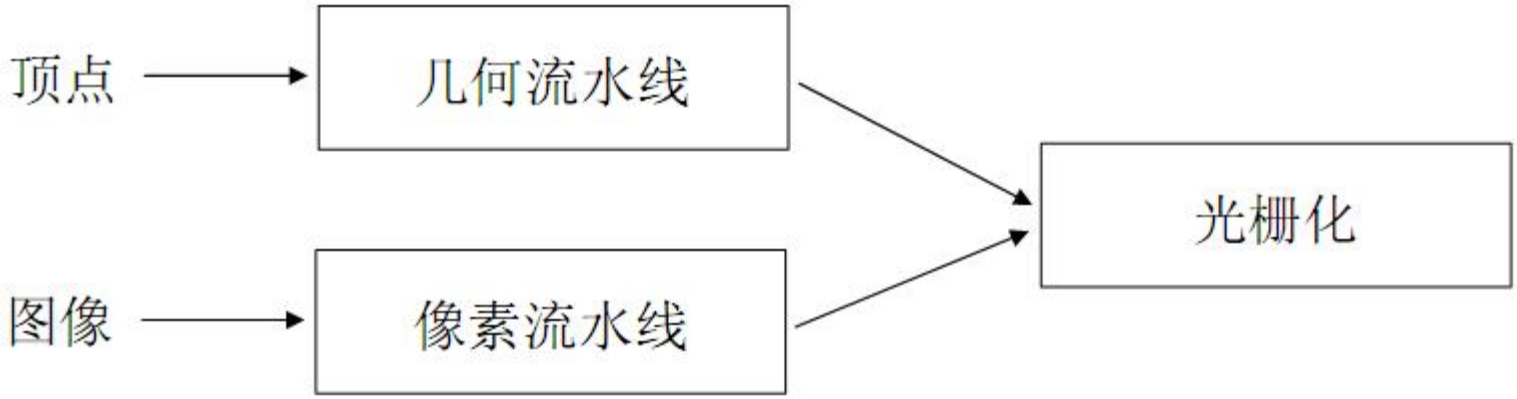
- 纹理(下方)是 256 × 256 的图像，它被映射到一个矩形上，经透视投影后的结果显示在上方



# 纹理映射与WebGL流水线

福州大学 FUZHOU UNIVERSITY

- 图像与几何分别经过不同的流水线，在光栅化时合二为一
  --复杂纹理并不影响几何的复杂性



# 纹理对象



- 创建一个纹理对象

```
var texture = gl.creatTexture();
```

- 绑定为二维纹理对象

```
glBindTexture(GL_TEXTURE_2D,texture);
```

# 纹理图像数组

- gl.TexImage2D(target,level,iformat,width,height,
  border,format,type,texelArray);

例如：

```
var texSize=64;
    numRows=9;
    numCols=8;
var myTexels=new Uint8Array(4*texSize*texSize);
```

gl.TexImage2D(GL_TEXTURE_2D,0,gl.RGBA,texSize,texSize,
0,gl.UNSIGNED_BYTE,myTexels);

这样WebGL就能够读取图像中的像素并把它存储在纹理内存中。

# 纹理图像数组



- 使用标准的Web格式的图像作为纹素：

```
var mTexels = new Image();
 image.src = “logo.gif”;
 gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL,true);
 gl.texImage2D(gl.TEXTURE_2D,0,gl.RGB,gl.RGB,
                    gl.UNSIGNED_BYTE,myTexels);
```

- HTML:

```
<img id = “logo”  src = “logo.gif”></img>
```

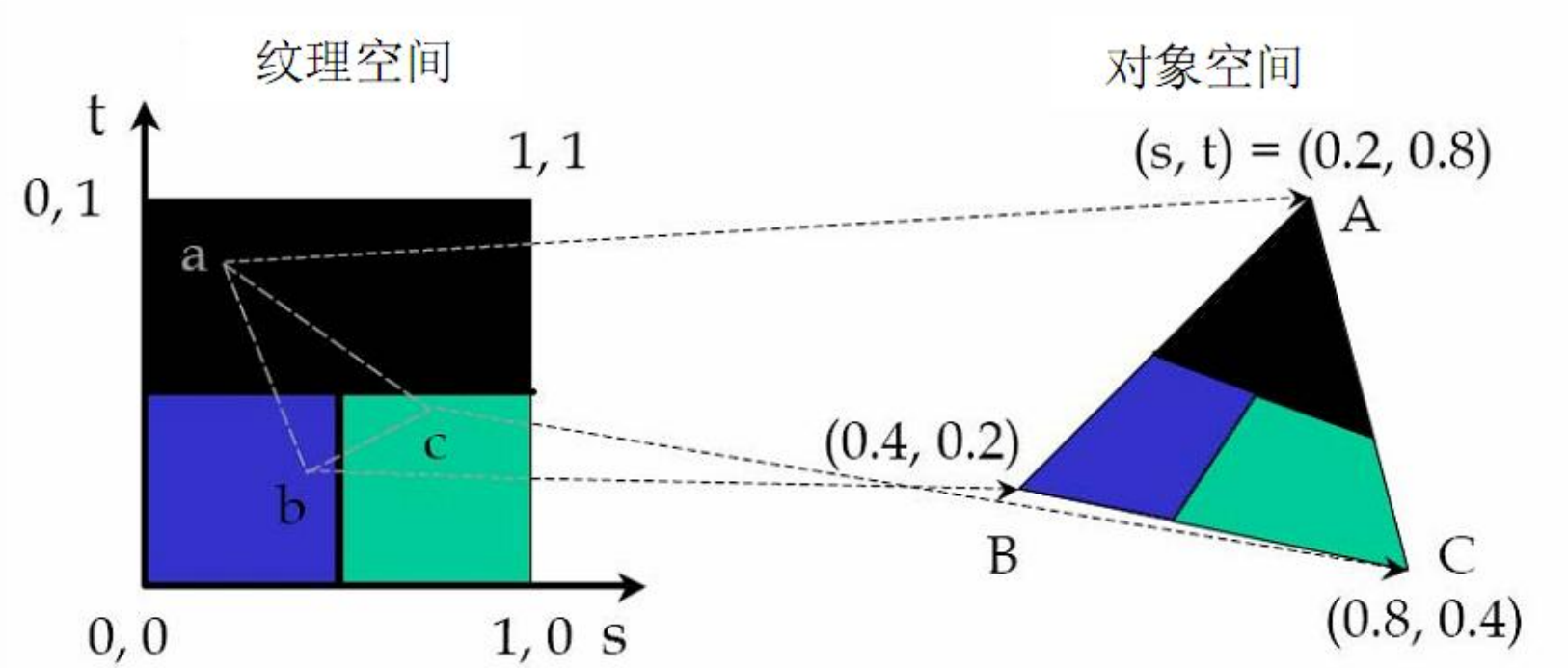
JS:

```
var image = document.getElementById(“logo”);
```

# 映射坐标和纹理采样器



- 纹理坐标




- 片元的纹理坐标由WebGL应用程序代码和着色器决定。

# 映射坐标和纹理采样器



- 最常用的方法是把纹理坐标作为顶点的属性
  - ✓类似顶点、颜色的方式来给定纹理坐标
  - ✓将纹理坐标传送到顶点着色器中
  - ✓顶点纹理坐标经过光栅化模块的插值计算得到片元的纹理坐标

```
GLfloat  tex_coord[N][2];
glGetAttribLocation(program,”texcoord”);
```

# 映射坐标和纹理采样器

福州大学 FUZHOU UNIVERSITY

- 纹理采样器变量： sampler1D、sampler2D
- 内置函数texture2D：可以从纹理图像提取像素值，赋值给内置变量gl_FragColor

- 使用一个uniform变量把WebGL应用程序代码中创建的纹理对象mytex与着色器关联起来：

  应用程序：

```
GLuint   tex_loc
tex_loc=glGetUniformLoction()
glUniform1i(tex_loc,0);
```

# 映射坐标和纹理采样器



- 片元着色器：

```
in vec2  st;
in vec4 color;
uniform  sampler2D  texMap;
void  main()
{
 gl_FragColor=color * texture2D(texMap,st);
 }
```

**vec4 texture2D(sampler2D sampler, vec2 coord)**

从 *sampler* 指定的纹理上获取 *coord* 指定的纹理坐标处的像素颜色。

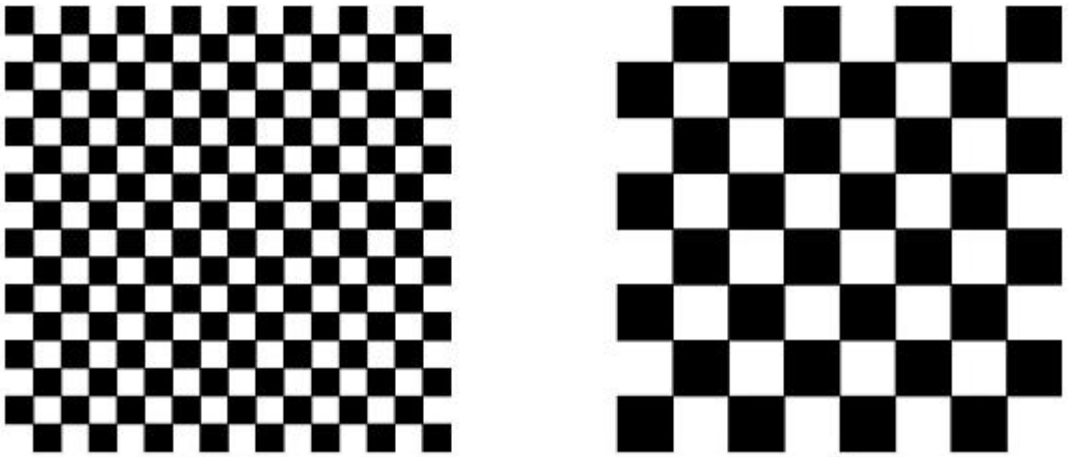
| | | |
|---|---|---|
| **参数** | sampler | 指定纹理单元编号 |
| | coord | 指定纹理坐标 |
| **返回值** | 纹理坐标处像素的颜色值，其格式由 gl.texImage2D() 的 *internalformat* 参数决定。表 5.9 显示了不同参数下的返回值。如果由于某些原因导致纹理图像不可使用，那就返回 (0.0, 0.0, 0.0, 1.0) | |

# 插值



- WebGL应用双线性插值从给定的纹理坐标中求出适当的纹理元素
- 可以只应用纹理的一部分

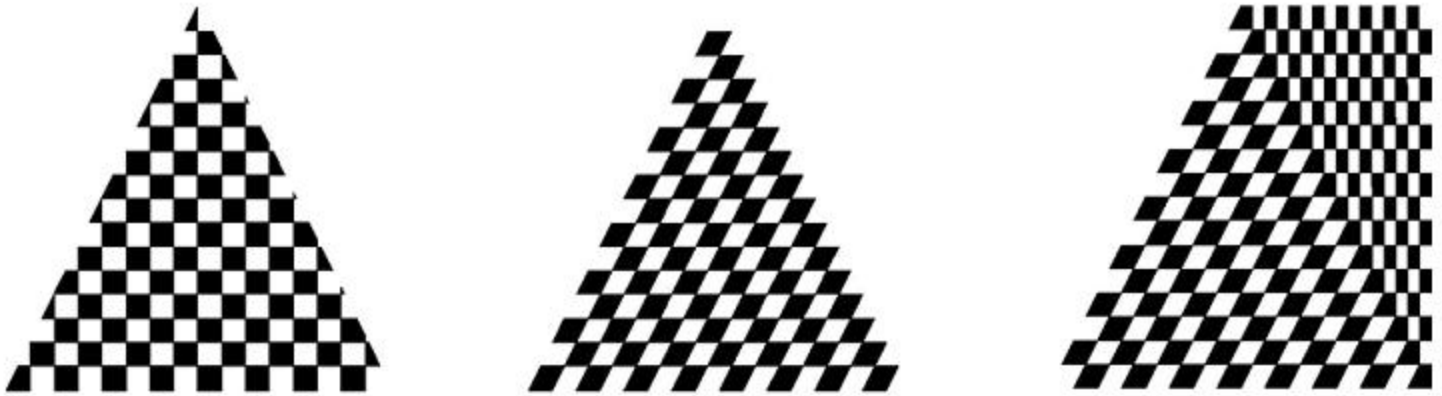
  --方法是只应用纹理坐标的一部分，如最大纹理坐标为(0.5,0.5)



(a) (b)

# 变形

- 对于四边形，从纹理坐标到纹理元素的对应是比较直接的
- 对于一般的多边形，WebGL需要确定一种方法，确定纹理坐标与纹理元素的对应
  - 可能会出现变形



# 纹理参数



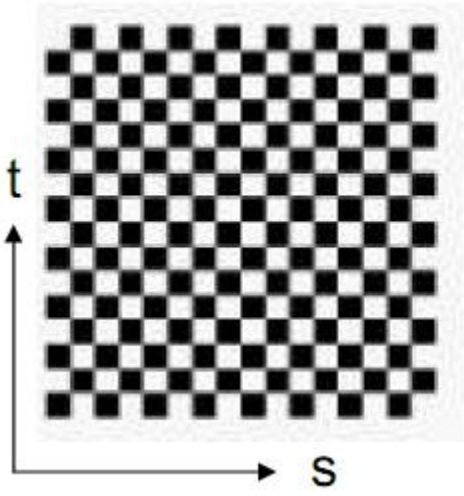
- WebGL中有许多办法确定纹理的使用方式
  - Wrapping参数确定当s, t的值超出[0,1]区间后的处理方法
  - 应用filter模式就会不采用点取样方法，而是采用区域平均方法
  - Mimmapping技术使得能以不同的分辨率应用纹理
  - 环境参数确定纹理映射与明处理的交互作用

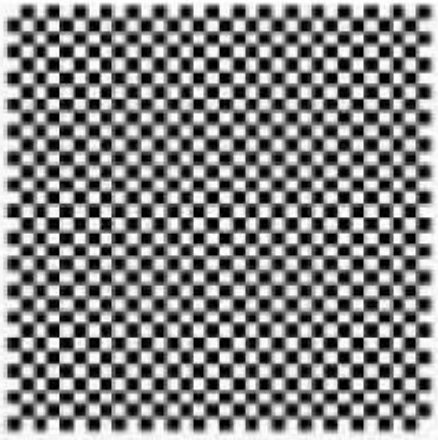
# Wrapping模式


- 截断：若s,t>1就取1，若s,t<0就取0
- 重复：应用s,t模1的值

gl.texParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_CLAMP)

gl.texParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT)



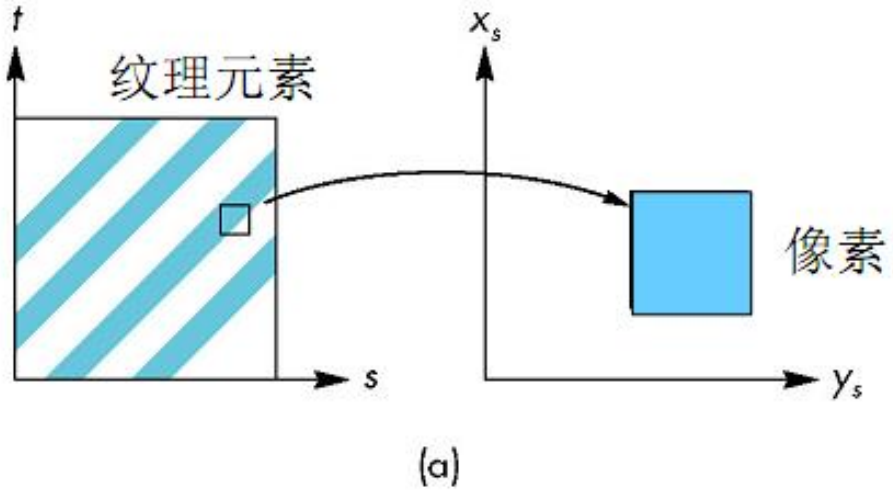



GL_REPEAT

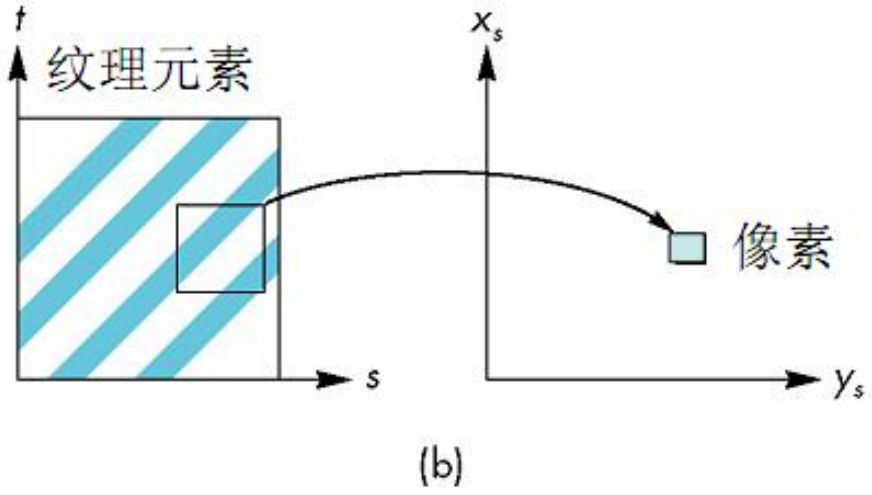
GL_CLAMP

# 放大与缩小

- 可以是多个纹理元素覆盖一个像素(缩小)，也可以是多个像素覆盖一个纹理元素(放大)



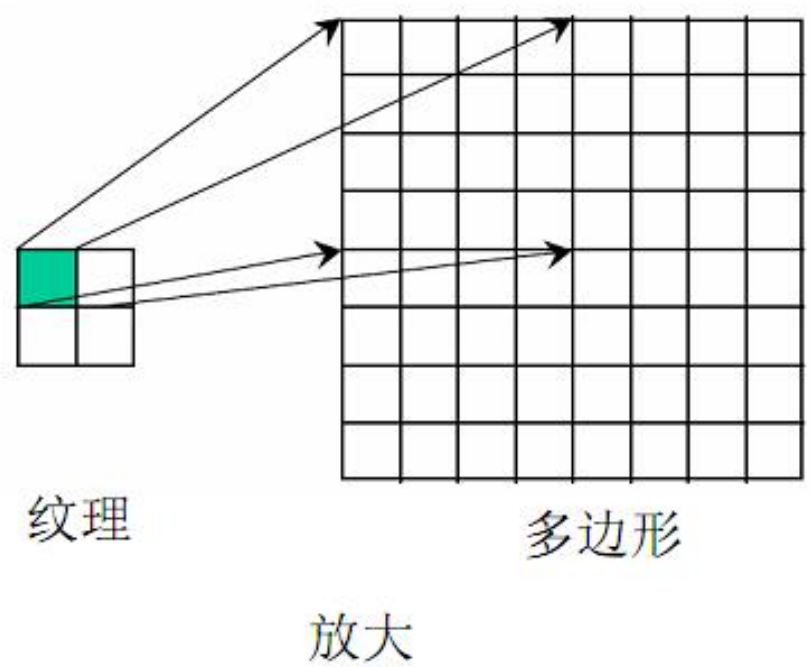
(a)

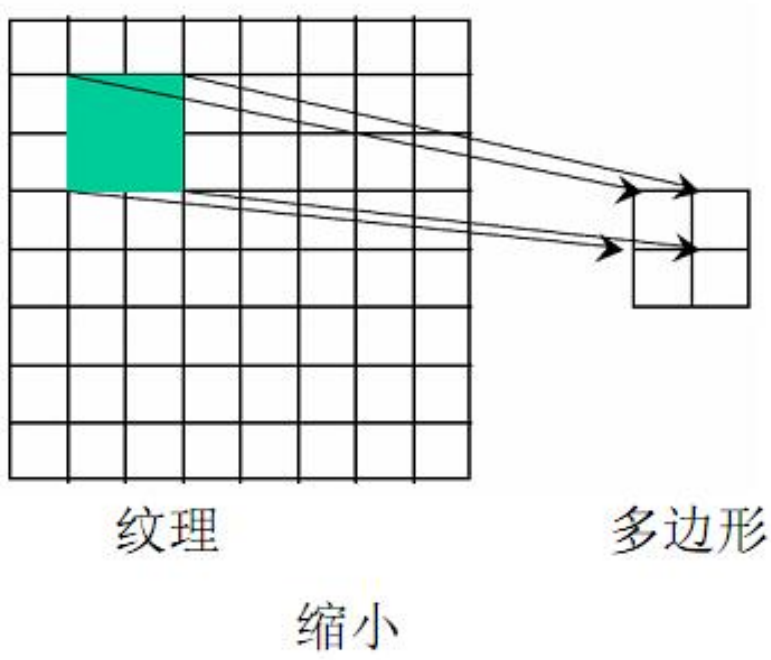


(b)

# 解决方法

- 可以应用点取样(最近纹理元素)或者线性滤波(2x2滤波)得到纹理值



# 滤波模式



- 模式指定

  glTexParameteri(target, type, mode)

  glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);

  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

- 注意在线性滤波中为滤波边界需要纹理元素具有额外的边界(border=1)

# 纹理的Mipmap

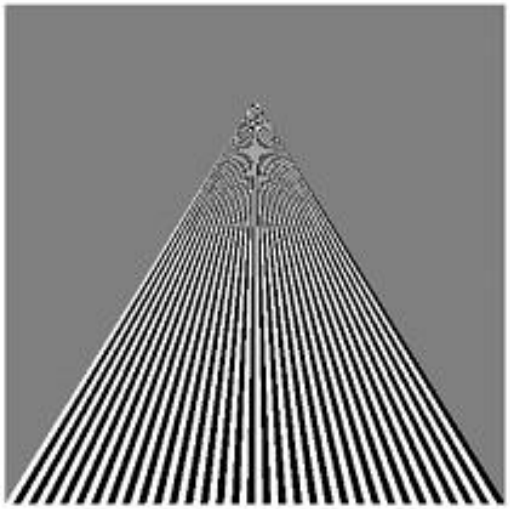


- Mipmap对纹理位图进行预先滤波，降低分辨率可以减小对于非常小的要加纹理的对象的插值误差
- 在纹理定义时声明mipmap的层次
  --glTexImage2D(GL_TEXTURE_2D, level, …);
- WebGL中提供了mipmap建立界面，可以从给定图像建立起所有的mipmap纹理
  --gl.generateMipmap(gl.TEXTURE_2D)

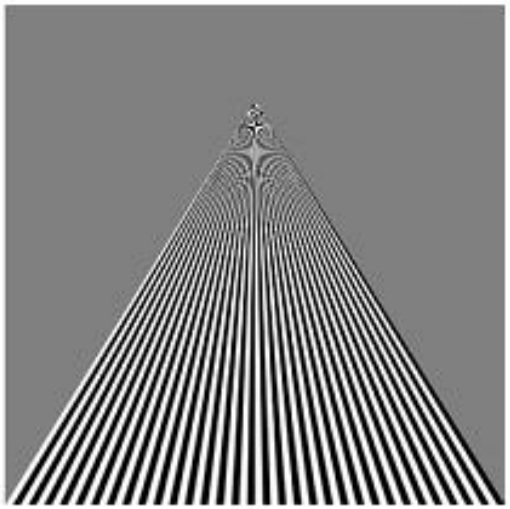
# 示例



点取样



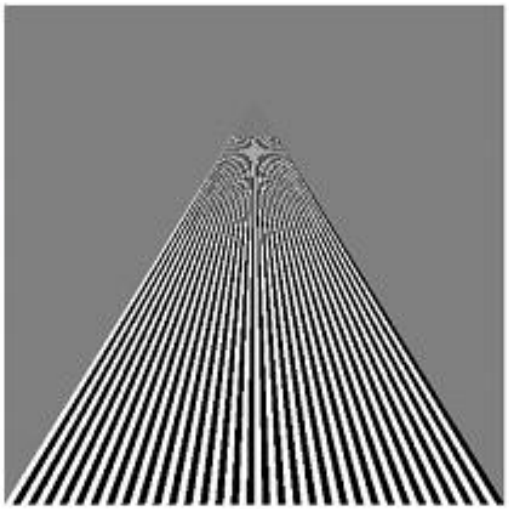
(a)

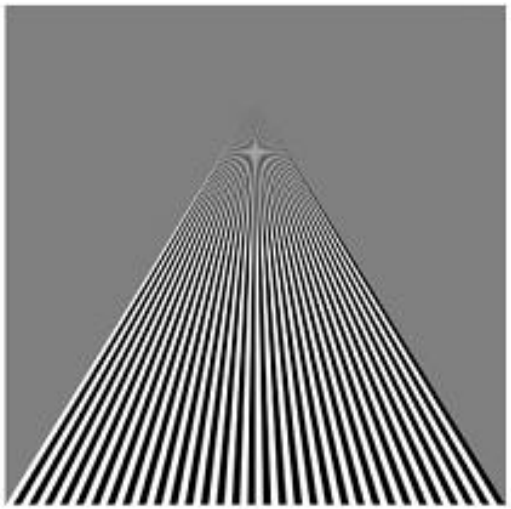


(b)

线性滤波

mipmapped
后的点取样





mipmapped
的线性滤波