2018-2019年度第二学期 00106501

计算机图形学

童 伟 华　管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

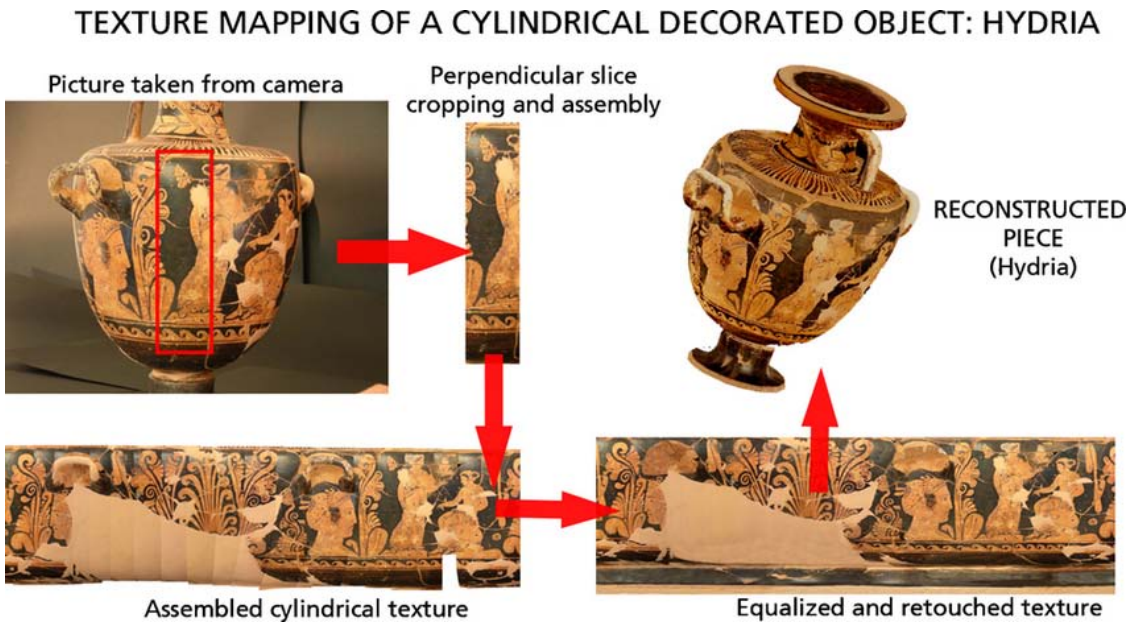
中国科学技术大学　数学科学学院

http://math.ustc.edu.cn/

第二节 纹理映射

几何建模的局限

- 虽然图形显示卡可以每秒钟显示多达一千万个多边形，但这个速度并不能满足模拟任何现象的要求，如何描述模型的细节？
  - 石头、树皮、皮肤等
  - 云、草、地貌等
  - 毛发、水波与火焰等

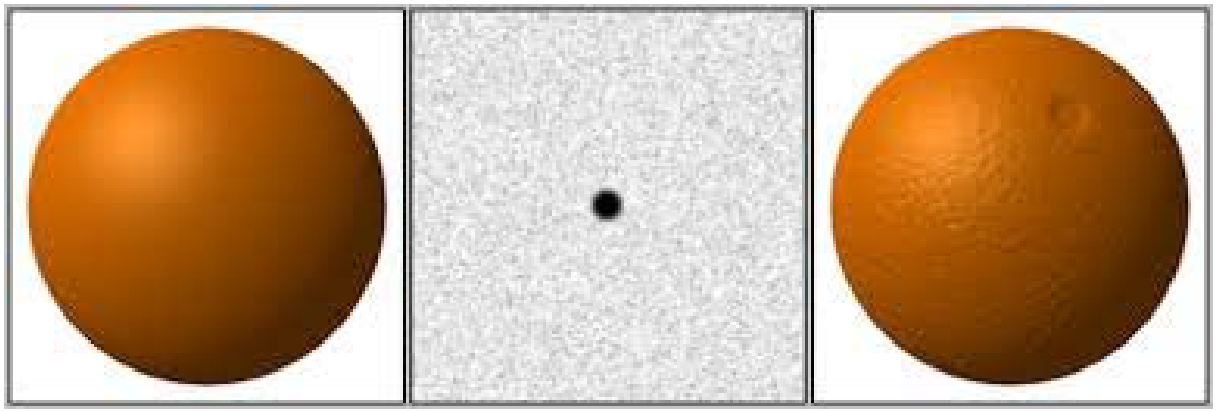


桔子的模型

- 考虑如何建立桔子（或其它水果）的模型
- 用着色的球表示桔子
  - 太简单
- 用更复杂形状代替球
  - 并没有强调曲面的特征（微凹，dimples）
  - 为了模拟所有的微凹，需要相当多的多边形

桔子的模型（续）

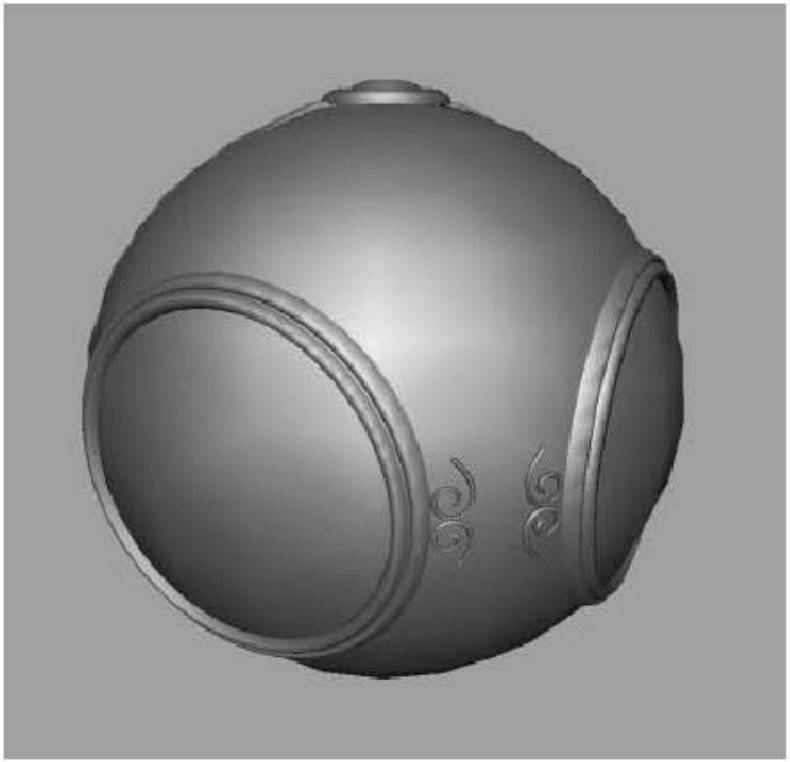
- 获取真实桔子的照片，扫描后把结果“粘贴”到简单的球模型上
  - 这个过程就是纹理映射
- 可能结果仍然不令人满意，因为所得曲面是光滑的
  - 需要改变局部形状
  - 凹凸映射



三种映射方法

- 纹理映射
  - 利用图像填充多边形
- 环境映射（反射映射）
  - 利用环境的图像进行纹理映射
  - 可以模拟高度镜面曲面
- 凹凸映射
  - 在生成显示结果的过程中可以改变法向量

纹理映射

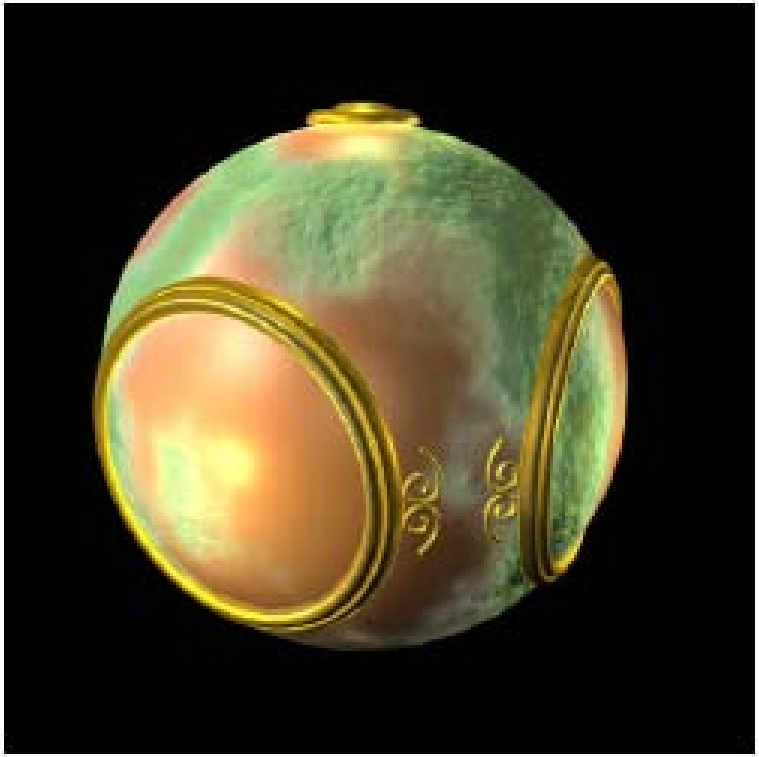


几何模型

纹理映射后

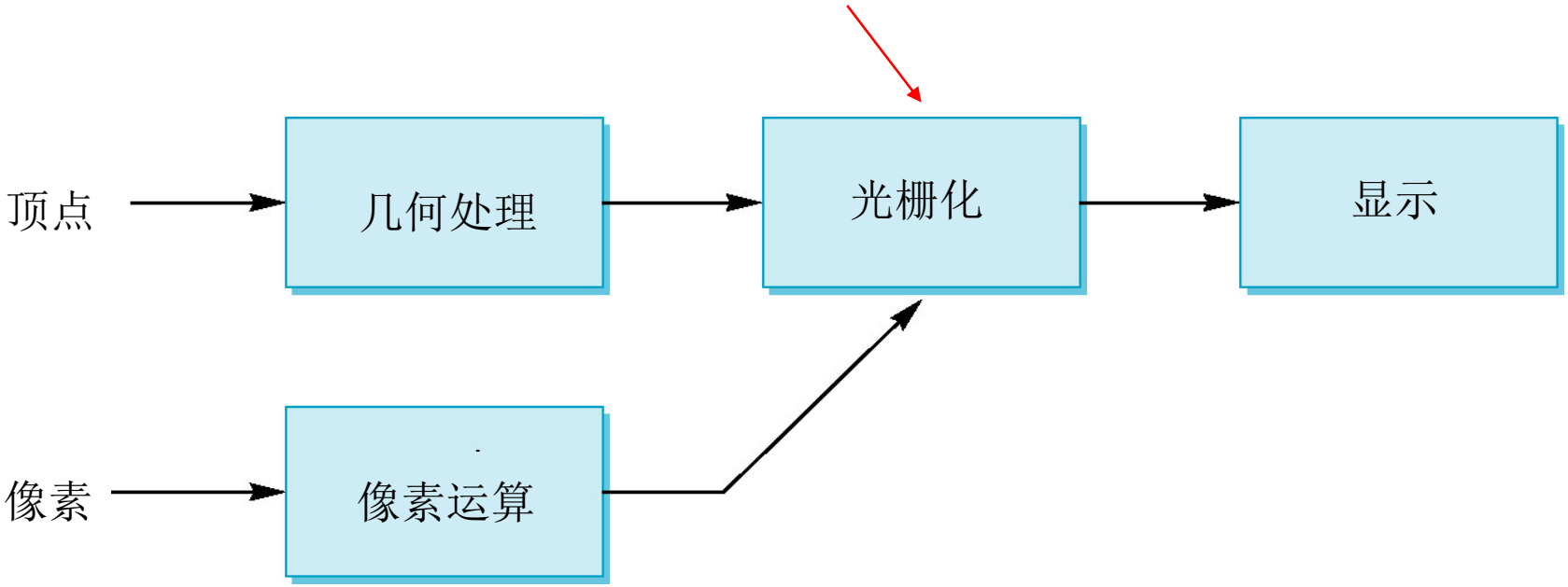
环境映射

凹凸映射



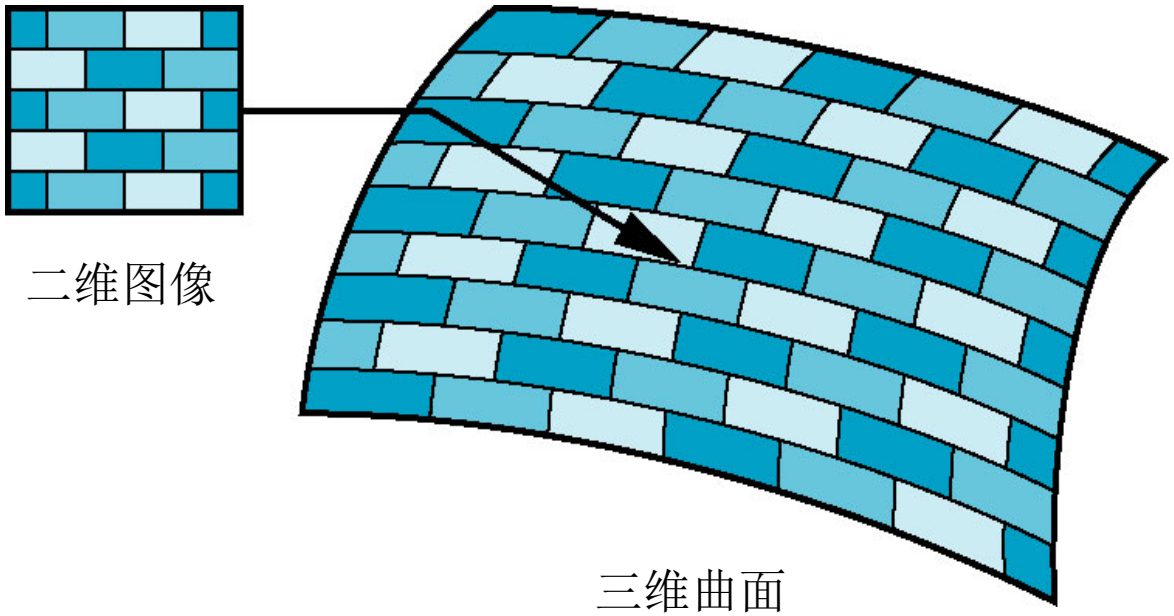
映射在什么地方进行？

- **映射技术是在输出流水线的最后阶段实现的**
  - 非常有效，因为在经过所有的操作后，减少了许多不必要的映射



很容易吗？

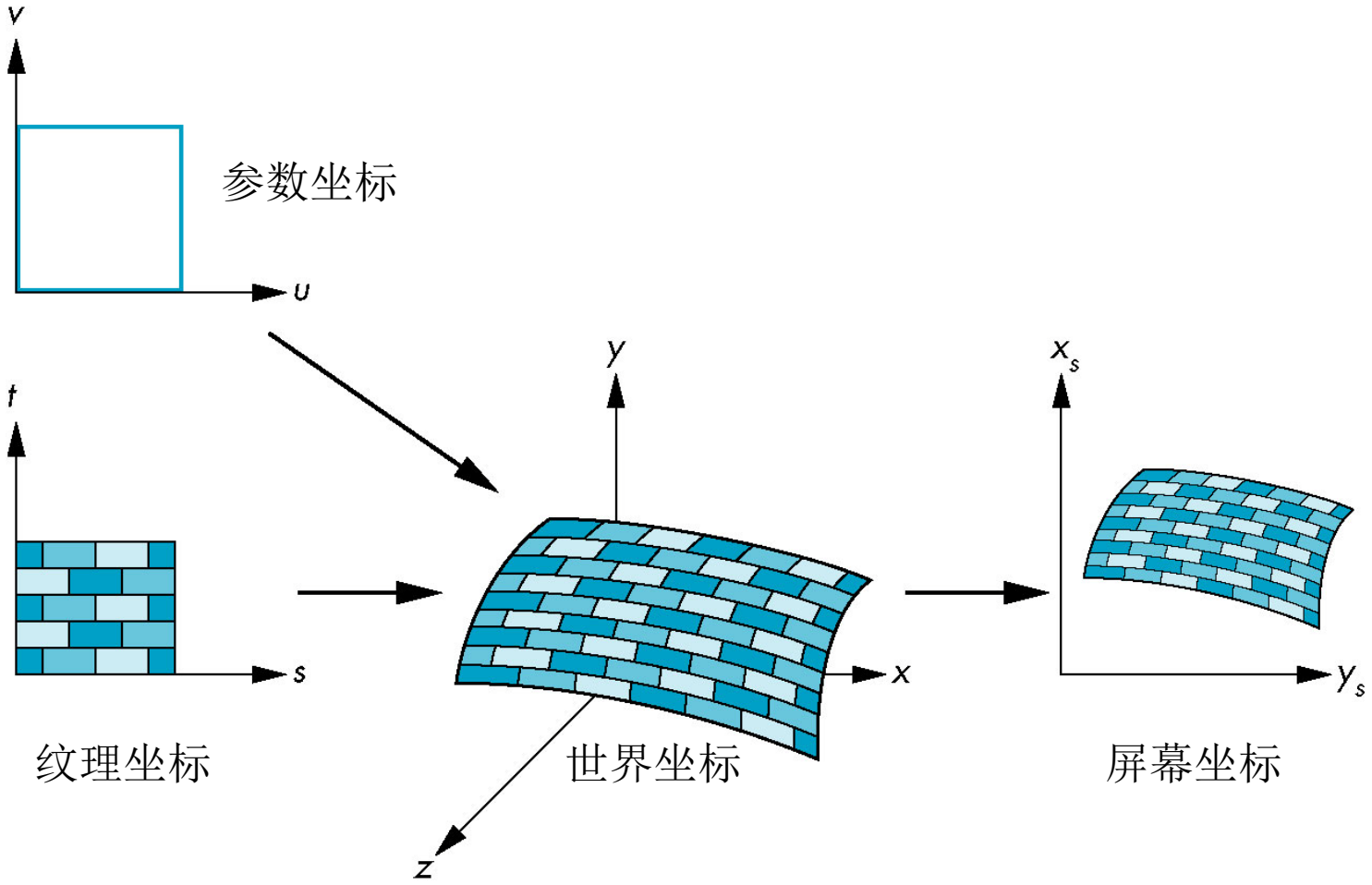
- 虽然映射的想法很简单，即把图像映射到曲面上，但由于这肘要用到三四个坐标系，因此实现起来并不容易



坐标系

- 参数坐标系
  - 可以用来建立曲面
- 纹理坐标系
  - 用来区别要被映射的图像上的点
- 世界坐标系
  - 从概念上说，就是映射发生的地方
- 屏幕坐标系
  - 最终图像生成的地方

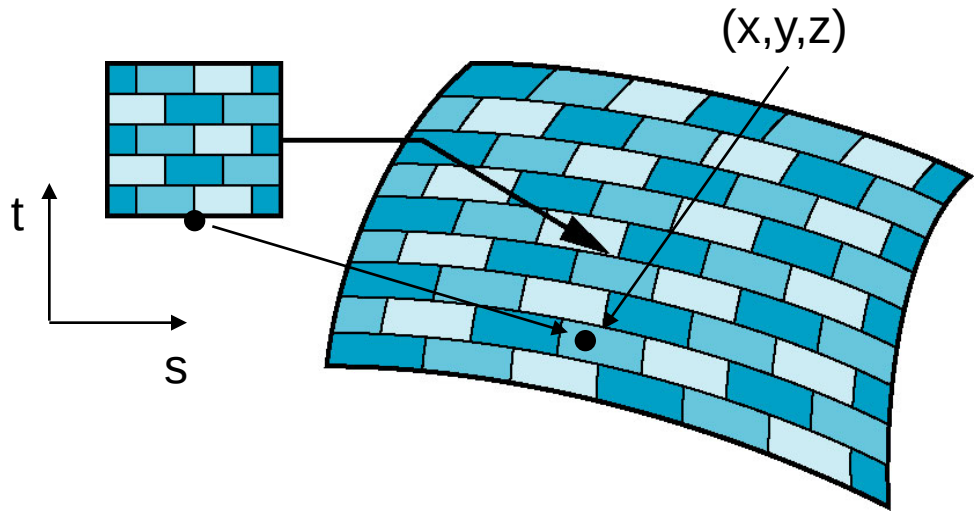
纹理映射框架



映射函数

- 基本问题就是如何定义映射
- 考虑从纹理坐标到曲面上一点的映射
- 直观地看，应当需要三个函数

  $x = x(s,t), y = y(s,t), z=z(s,t)$
- 虽然从概念上讲，最终必定用到上述的函数，但实际采用的是却是间接的方法

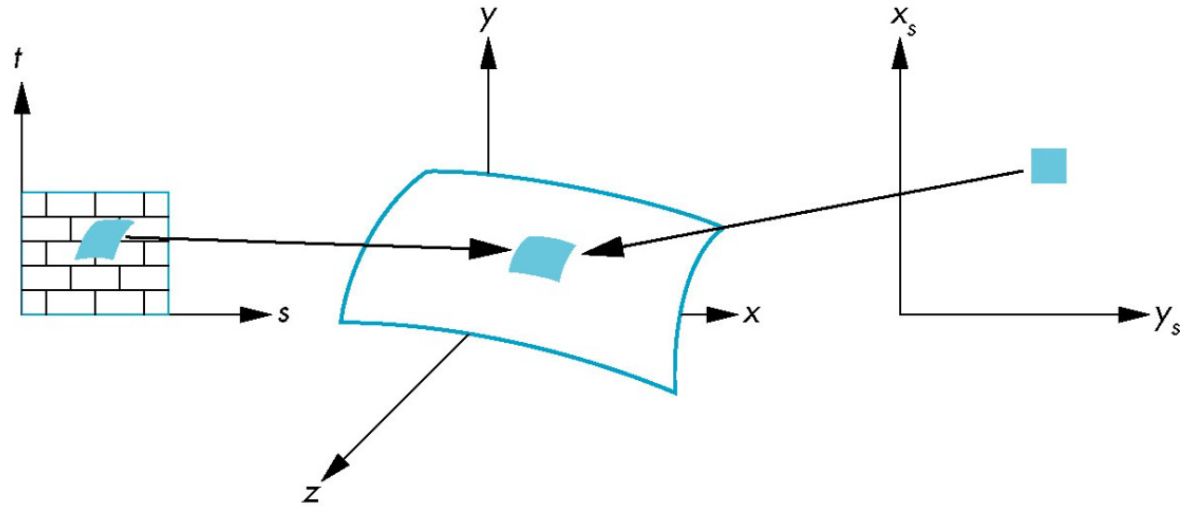


逆映射

- 我们需要的是逆向操作
  - 给定一个像素，我们想知道它对应于对象上的哪个点
  - 给定对象上的一个点，我们想知道它对应于纹理中的哪个点
- 此时需要如下形式的映射
  - s = s(x,y,z), t=t(x,y,z)
- 这样的函数一般是很难求出来的

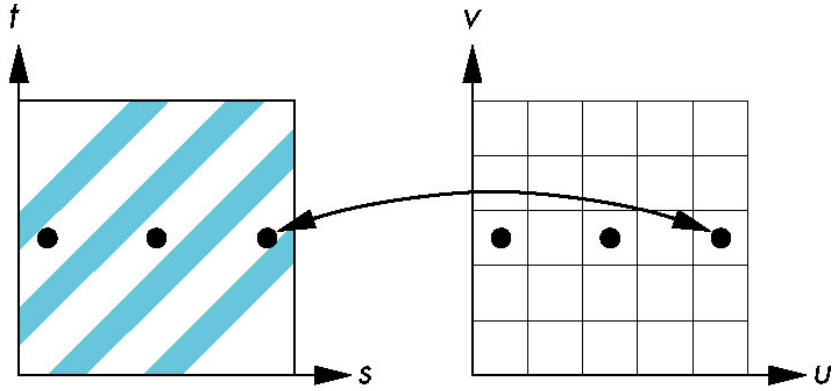
实际困难

- 假设要计算中心在(x_s, y_s)的一个矩形像素的颜色，中心点对应于对象上的点(x,y,z)
- 如果对象是弯曲的，那么矩形像素的原像是一个曲边四边形
- 这个曲边四边形在纹理上的原像才是对当前矩形像素的颜色有贡献的纹理元素



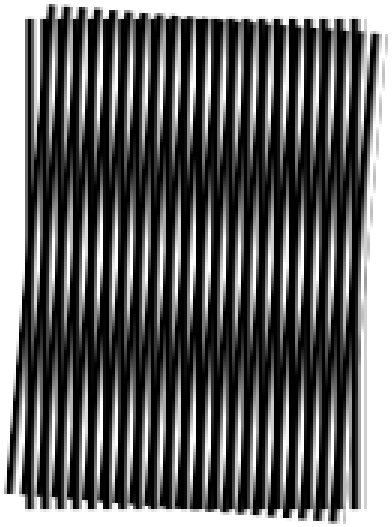
解决方法

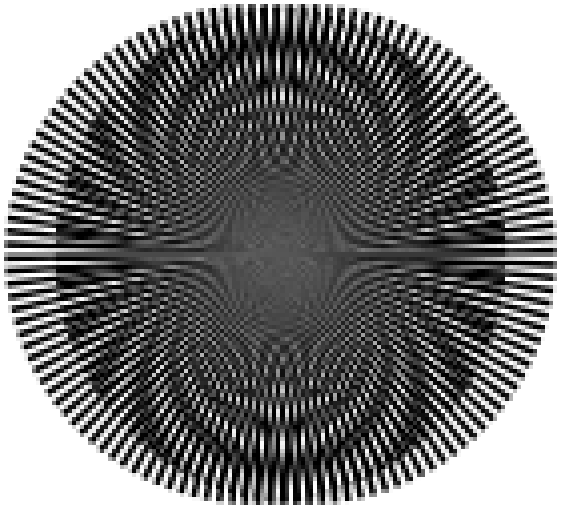
- 先不管如何定义逆映射，只考虑如何确定当前像素的颜色（或者说明暗效果）
- 一种方法是用当前像素块的原像的中心对应的纹理元素的颜色
  - 走样（aliasing）



Moiré效果

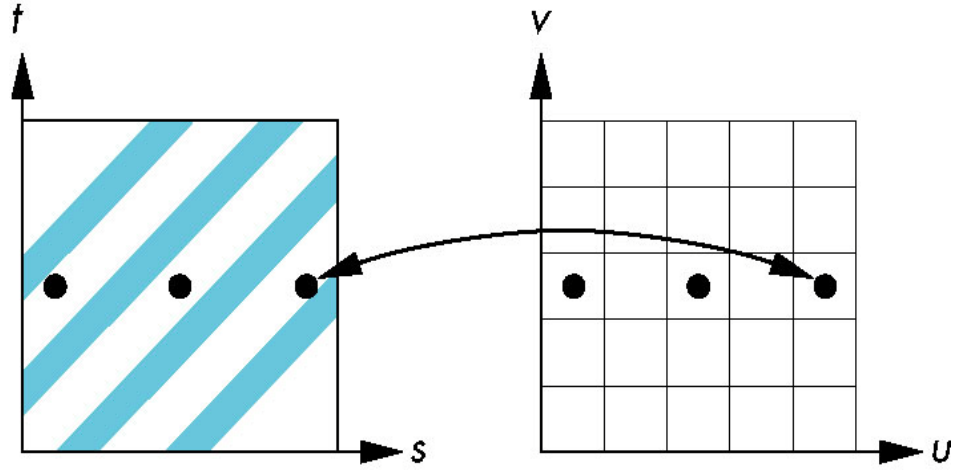
- 当不考虑像素是有一定的大小时，在图像中会导致moiré效果（莫列波纹，又译为摩尔纹、莫尔条纹、叠纹、水状波纹），是一种在栅栏状条纹重叠下所产生的干涉影像
  - 例如旋转某些有规律的图就会出现这种波纹





“更好”的方法

- 用在纹理原像上对应区域的明暗效果的平均值赋给当前像素
  - 也不是完美的
  - 右图经上述处理后也得不到所需要的结果
  - 对于规则纹理，这种效果非常明显
- 问题根源：帧缓存和纹理都只有有限分辨率，进行采样后，必然会产生走样现象



映射的确定

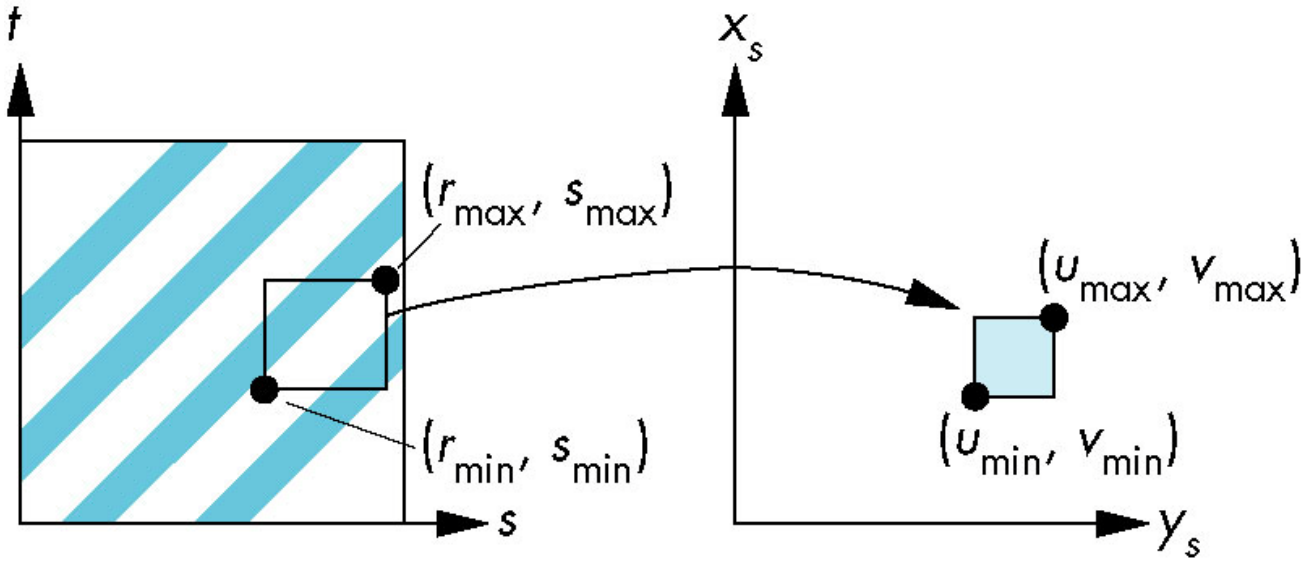
- 考虑由参数方程定义的曲面

$$p(u,v)=(x(u,v),y(u,v),z(u,v))$$

- 此时通常采用如下形式从纹理元素对应到曲面上的点

$$u = as + bt + c,\ v = ds + et + f$$

  - 只要$ae \neq bd$，上述映射是可逆的



$$u = u_{\min} + \frac{s - s_{\min}}{s_{\max} - s_{\min}}(u_{\max} - u_{\min})$$

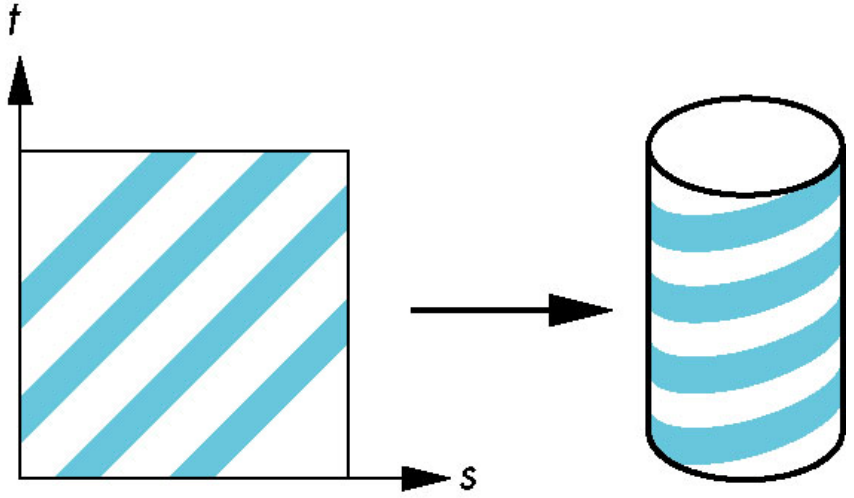
$$v = v_{\min} + \frac{t - t_{\min}}{t_{\max} - t_{\min}}(v_{\max} - v_{\min})$$

方法的特点

- 很容易应用
- 没有考虑曲面的弯曲
  - 为了填充曲面，纹理在不同方向进行了不同拉伸

两步映射

- 解决映射问题的另外一种方法
- 首先把纹理映射到一个简单的中间曲面上
- 例如：映射到圆柱上



圆柱映射

- 假设纹理坐标在单位正方形$[0,1]^2$内变化，圆柱高h，半径r
- 那么圆柱的参数方程为

  $x = r\cos(2\pi s), y = r\sin(2\pi s), z = ht$
- 从纹理坐标到圆柱面上没有变形
- 适合于构造与无底的圆柱面拓扑同构的曲面上的纹理

球映射

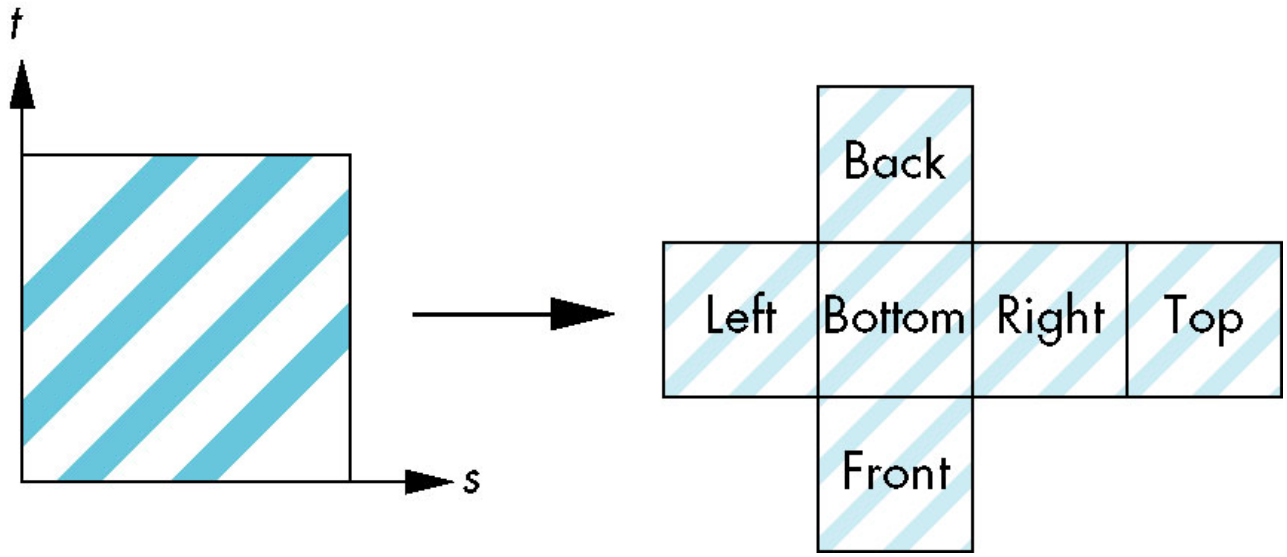
- 球的参数方程

  $x = r\cos(2\pi s), y = r\sin(2\pi s)\cos(2\pi t), z = r\sin(2\pi s)\sin(2\pi t)$

- 类似于地图绘制中的映射
  - 肯定有变形
- 用在环境映射中

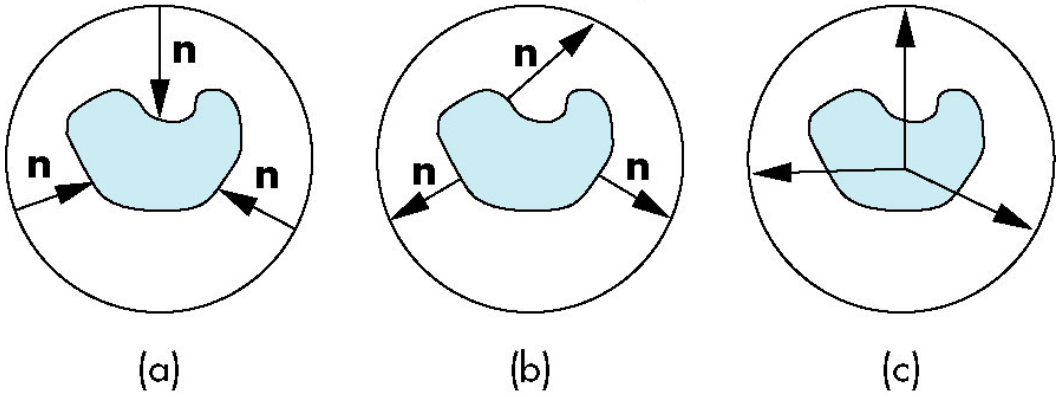
立方体映射

- 适合应用于正交投影
- 也用在环境映射中



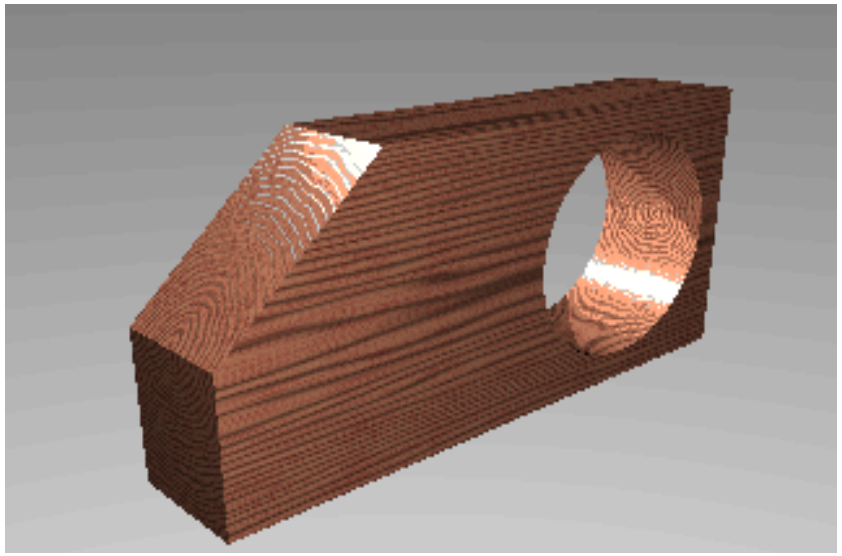
两步映射中的第二个映射

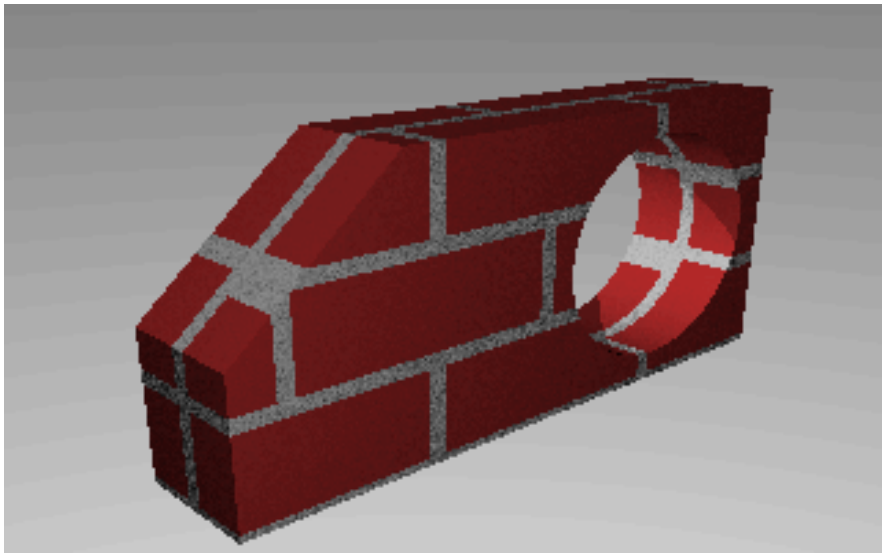
- 从中间对象到实际对象的映射
  - 从中间形状的法向到实际对象
  - 从实际对象的法向到中间形状
  - 从中间形状的中心开始的向量



其它类型的纹理

- 立体纹理

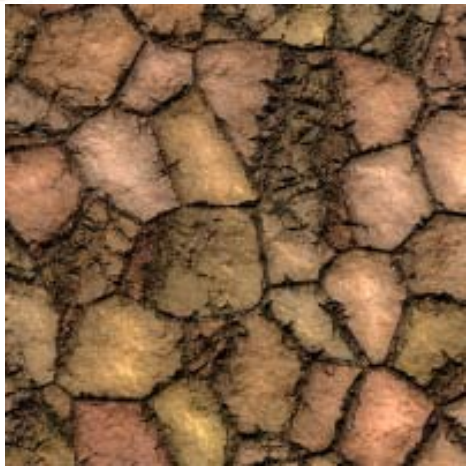




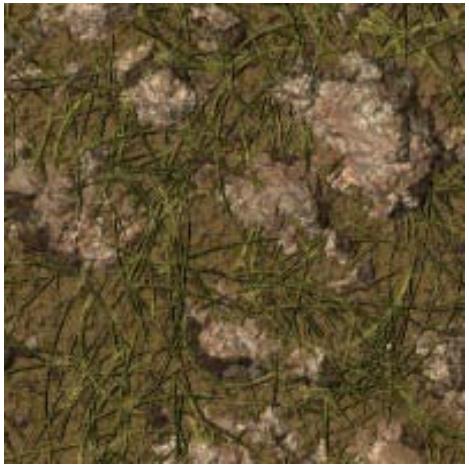
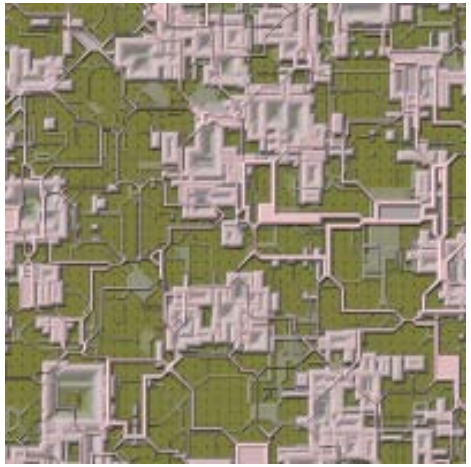
程序生成的纹理

- Procedural texture

更多的例子





Thanks for your attention!