



中山大學
SUN YAT-SEN UNIVERSITY



国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

计算机图形学

纹理

陶钧

taoj23@mail.sysu.edu.cn

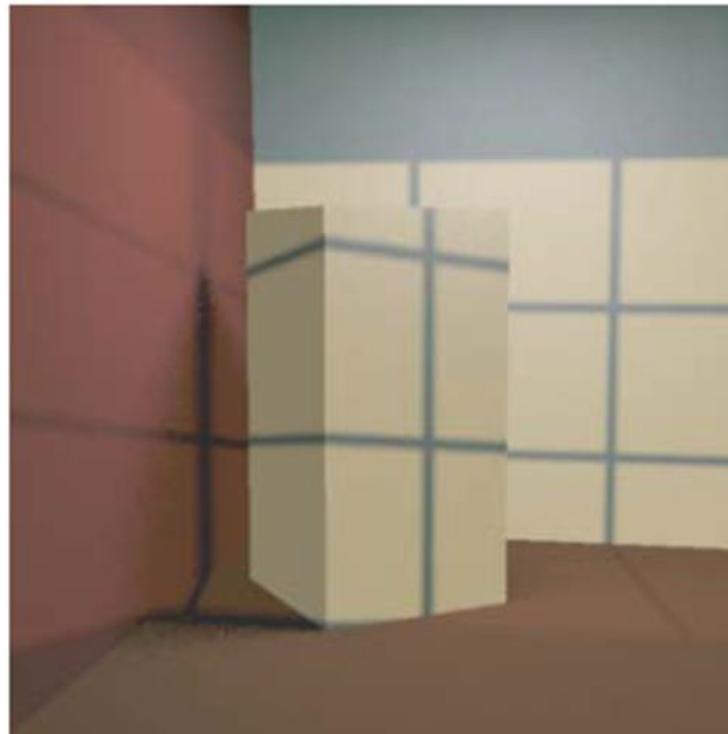
中山大学 数据科学与计算机学院
国家超级计算广州中心

- 纹理简介
- 纹理贴图方法
- 凸凹贴图与移位贴图
- OpenGL中使用纹理

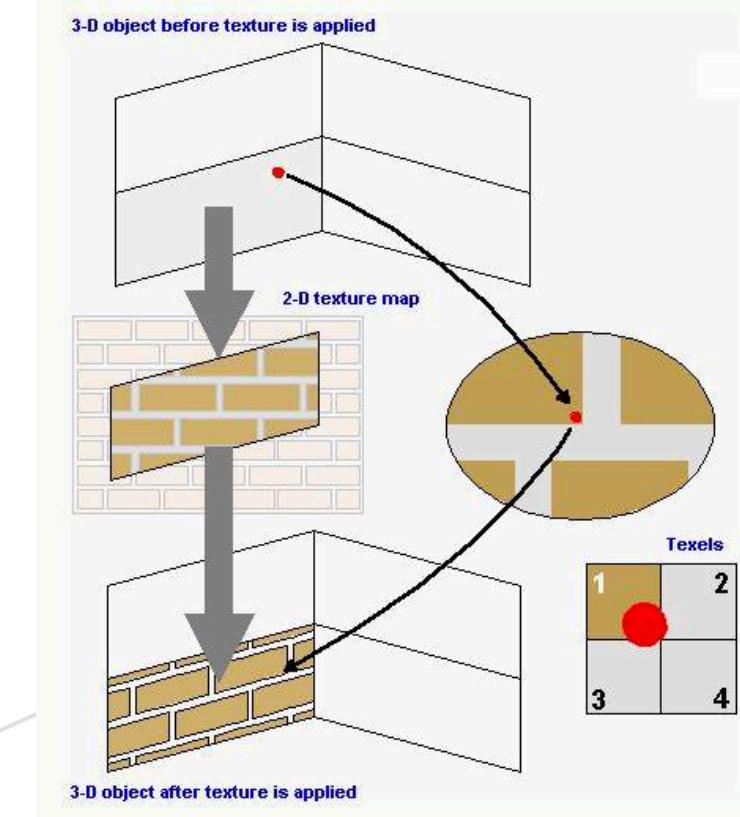


● 此前方法的局限性

- 指定每个顶点的颜色，而多边形内部像素颜色则从顶点颜色中插值得到
- 即使对于简单的几何形状，为表现复杂图案也需要复杂的网格
 - 如，图中的墙面，纸张上的图案等



Generated with Blue Moon Rendering Tools — www.bmrt.org



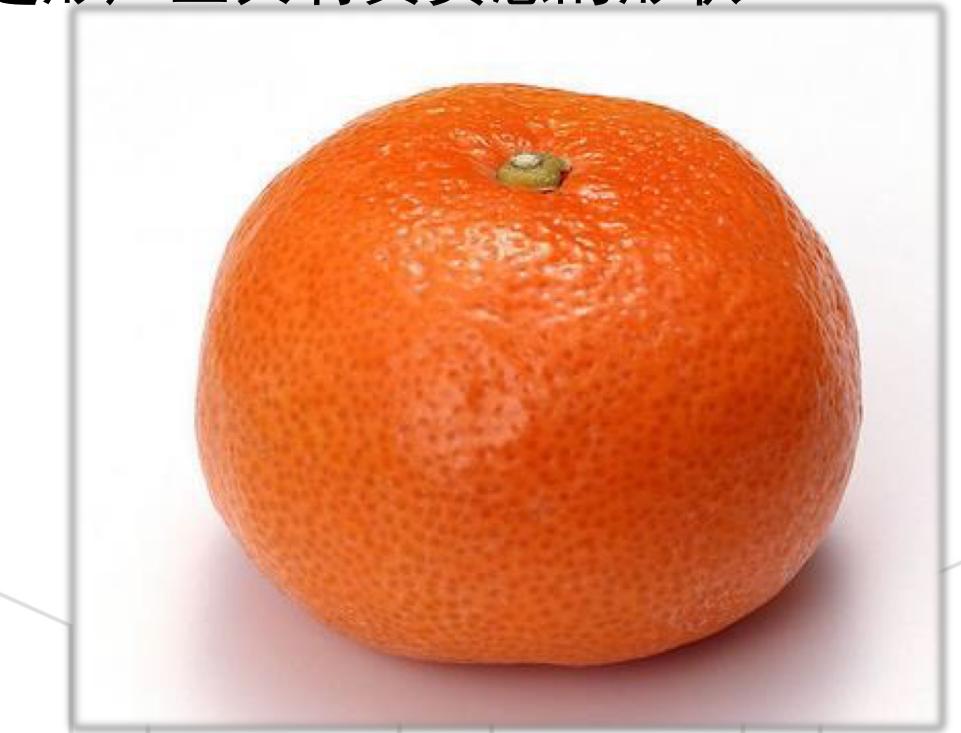
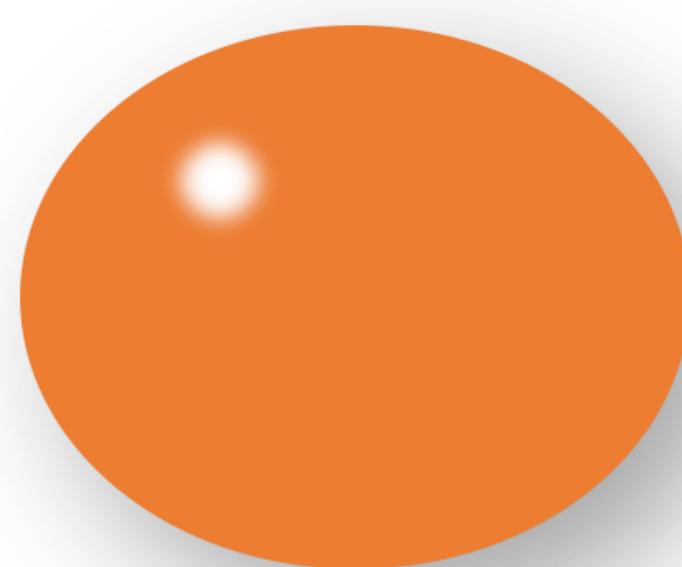
● 此前方法的局限性

- 即使当前显卡已经可以在一秒内渲染上千万个多边形，依然难以通过物理系统模拟自然界的所有现象/物体
 - 云，草，树叶，头发，不规则的陆地表面，水，火，等



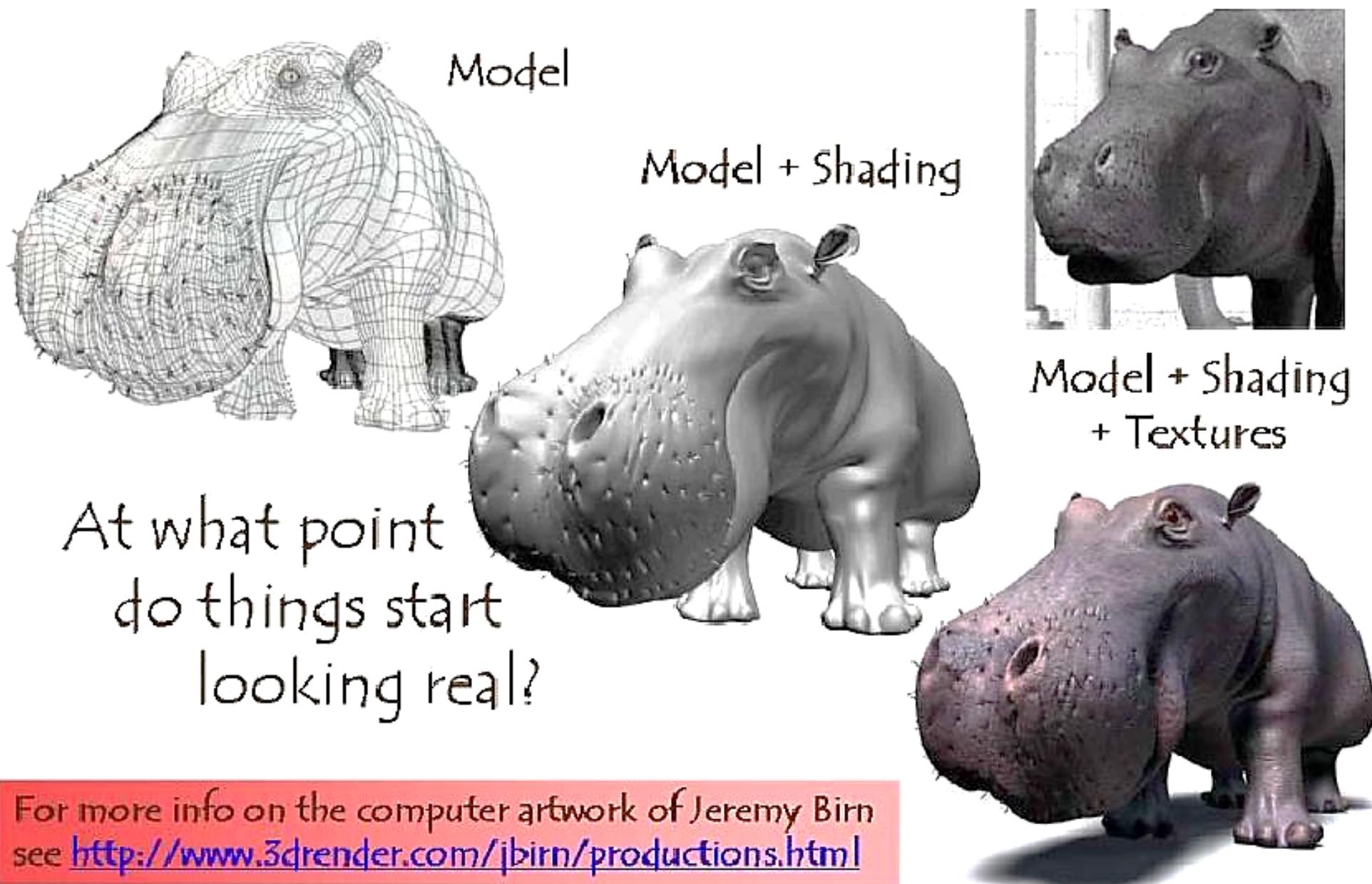
● 图像的真实感从何而来？

- 对橘子（或其他水果）建模
 - 使用一个简单的椭球体表示橘子？
 - 不真实！
 - 使用表面凹凸不平的复杂形状？
 - 需要极大数量的多边形产生具有真实感的形状



为什么要使用纹理？

- 图像的真实感从何而来？



- 纹理是曲面上属性的变化

- 颜色，法向量，对光的反射特性，位置偏移，等
- 计算机为产生具有真实感的图片，必须具备这些复杂细节
- 由于计算中产生的巨大开销，使用几何建模及物理模拟难以完整描述这些细节
- 纹理映射以较低的开销有效地“伪造”近似的表面细节

- 纹理贴图 (texture mapping)

- 将纹理通过变换“贴”至三维物体表面上
- 从函数定义域到三维物体表面的映射
 - 定义域可以为一维，二维，或三维
 - 函数可表示为数组或解析式

- 纹理简介
- 纹理贴图方法
- 凸凹贴图与移位贴图
- OpenGL中使用纹理



- 纹理贴图 (texture mapping)
- 环境/反射贴图 (reflection mapping)
- 凸凹贴图 (bump mapping)



● 参数坐标

- 三维曲面的二维参数化形式

● 纹理坐标

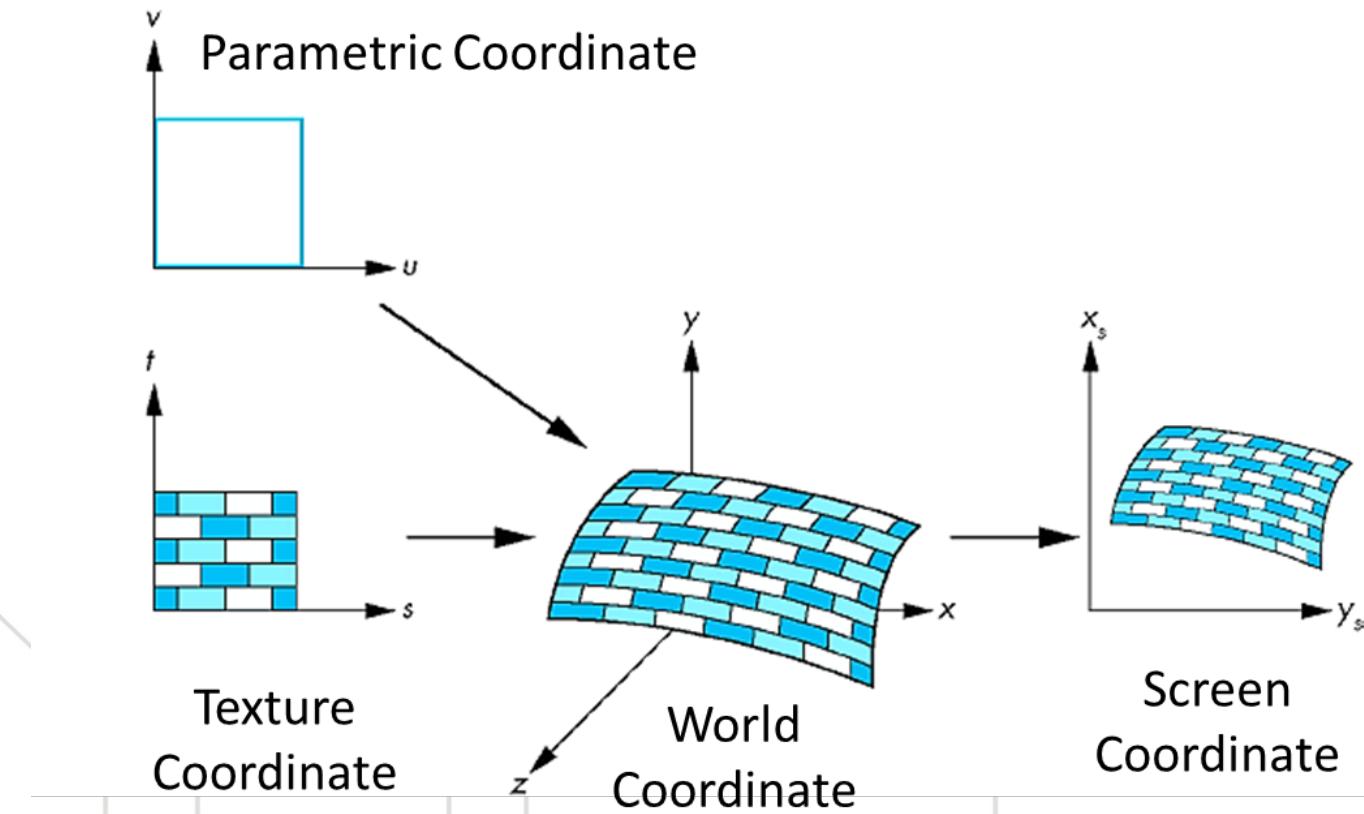
- 用于定位纹理中的具体纹理元素 (texel)

● 世界坐标

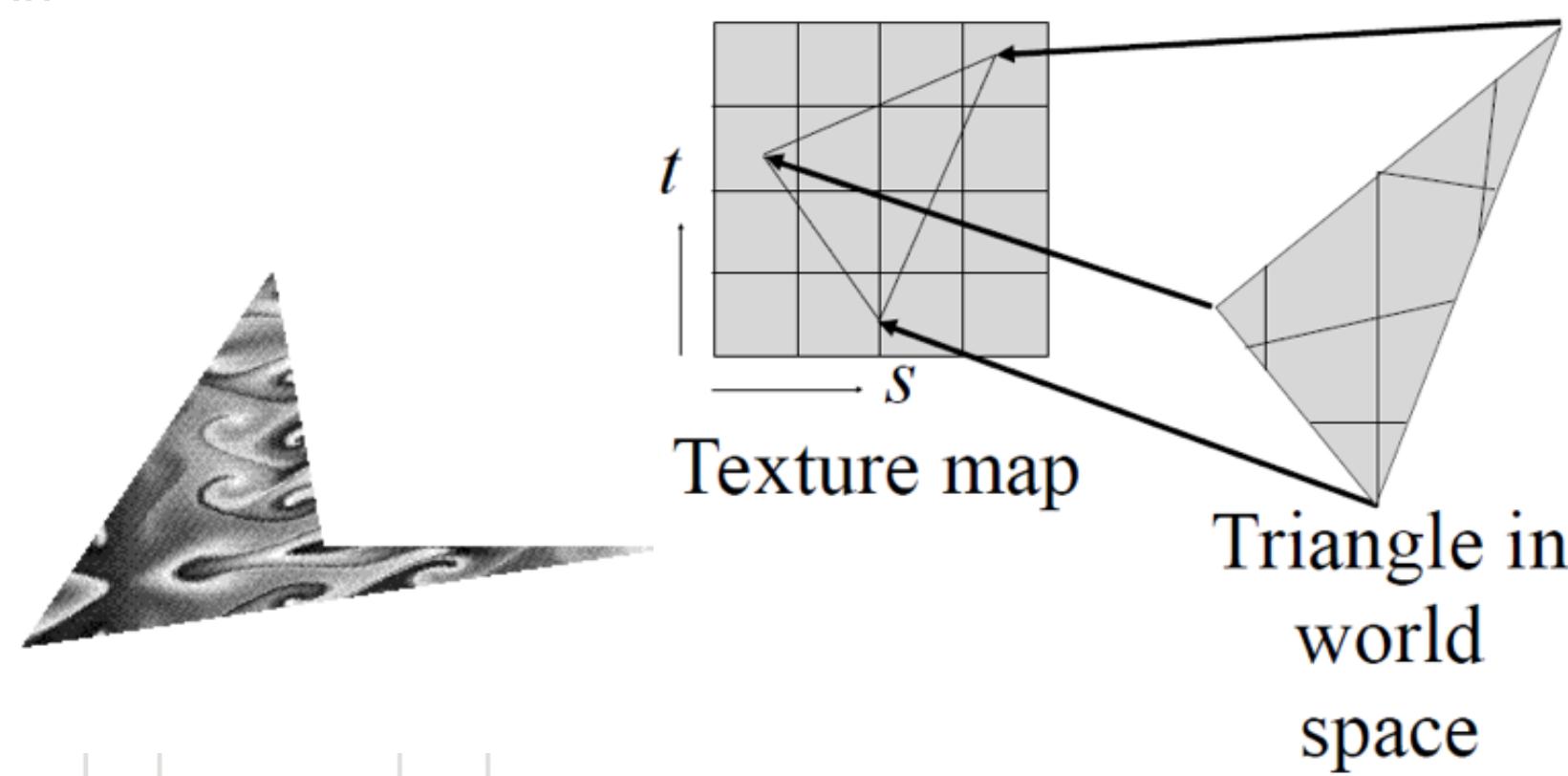
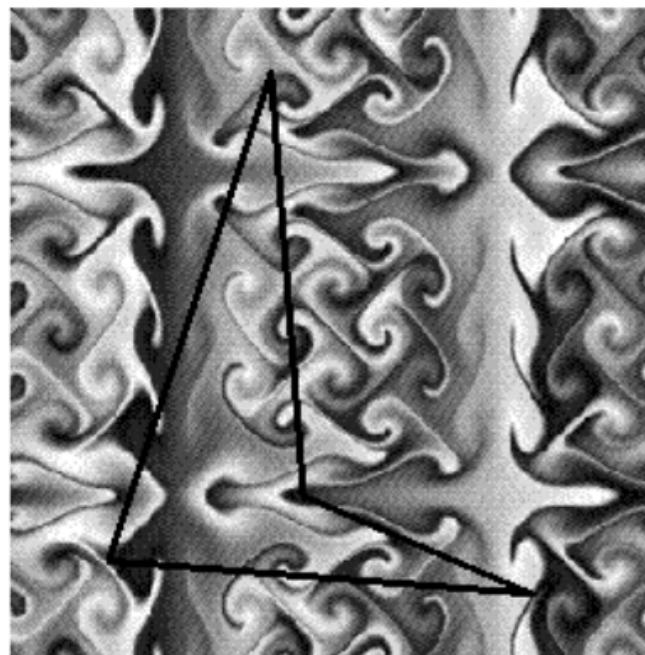
- 三维物体所处的空间

● 屏幕坐标

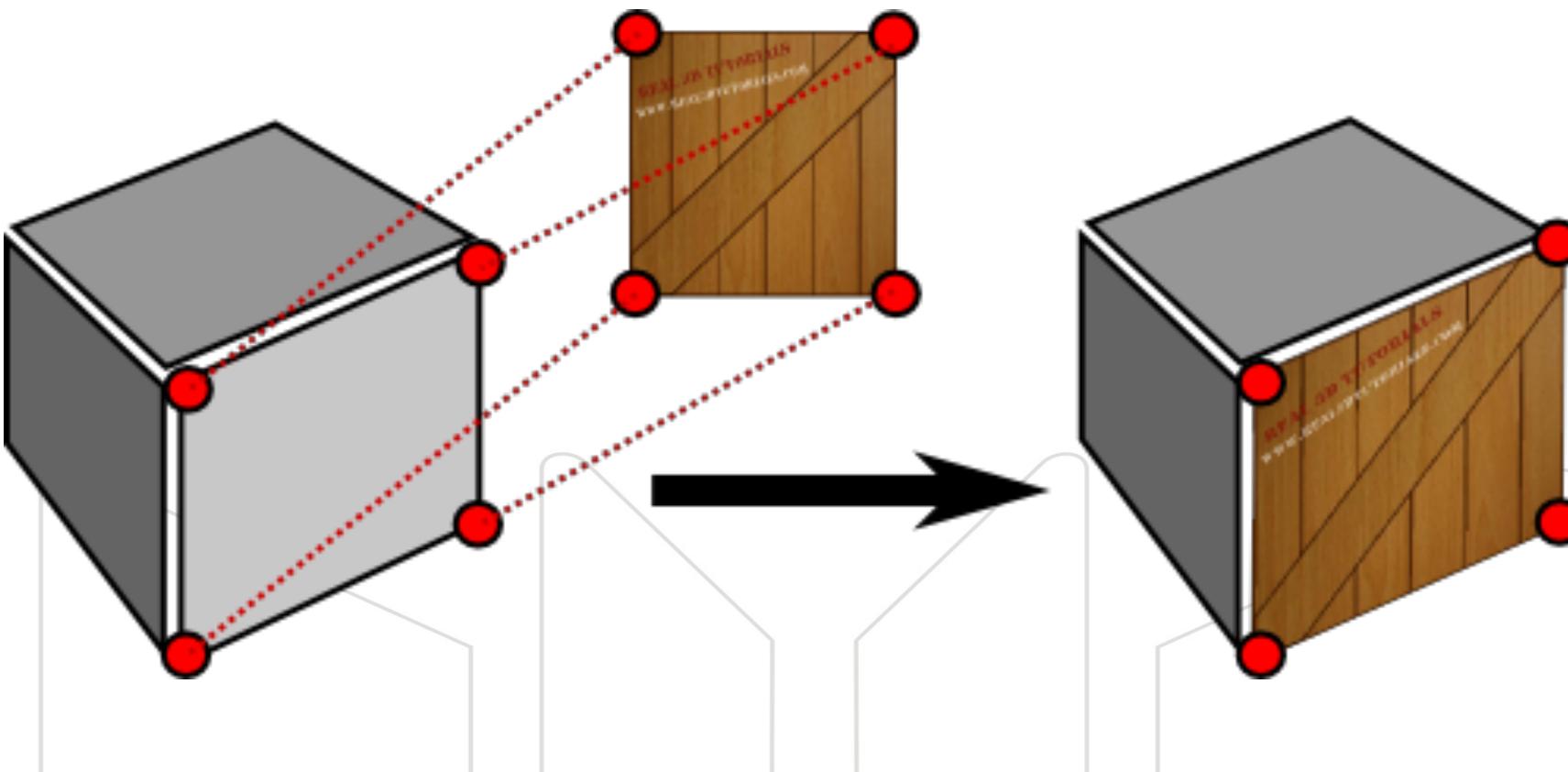
- 最终成像空间



- 指定顶点的纹理坐标，多边形内部对纹理坐标进行线性插值
 - 纹理坐标对应二维纹理图像中的纹理元素
 - 现在也有三维纹理图像
 - 顶点纹理坐标所包围的纹理图像部分将应用于多边形
 - 纹理将被拉升或压缩以适应多边形的大小及形状



- 多边形纹理贴图建立二维纹理到二维表面的对应关系
 - 尽管二维表面处于三维空间中
 - 使用扫描算法对纹理坐标进行插值，获取多边形内部像素所对应的纹理元素，从而决定该像素的特定属性取值（通常为颜色）



- 从纹理上的方块到曲面上的方块

- 只需将纹理上的方块范围

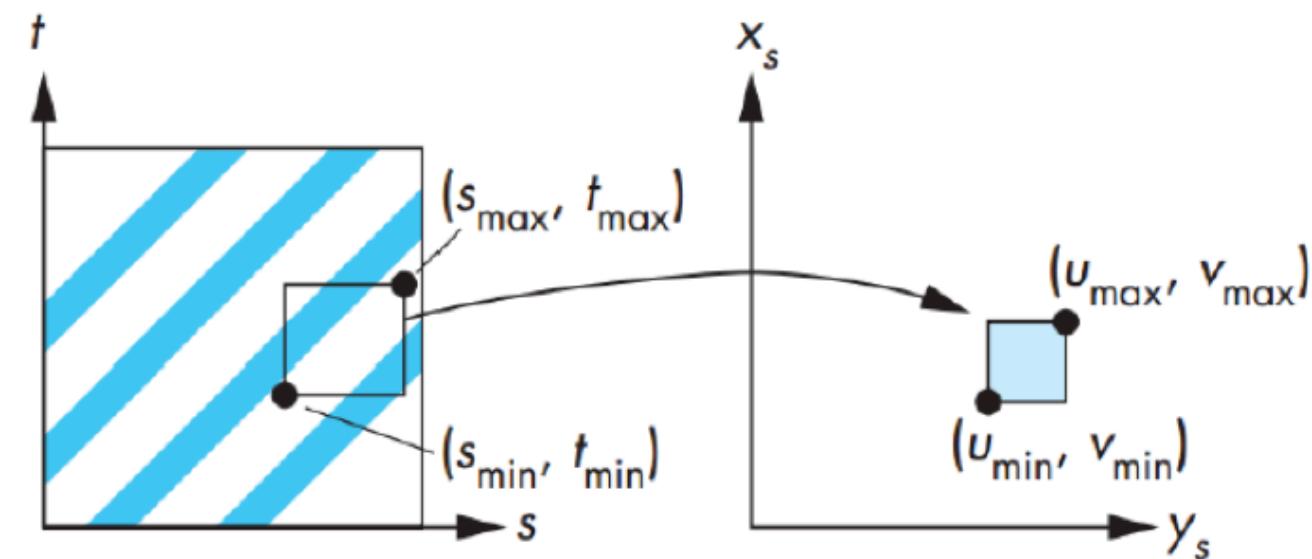
- $[(s_{min}, t_{min}), (s_{max}, t_{max})]$

- 映射至曲面上的方块范围

- $[(u_{min}, v_{min}), (u_{max}, v_{max})]$

- 简单易行

- 没有考虑曲面的曲率等形状信息



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

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



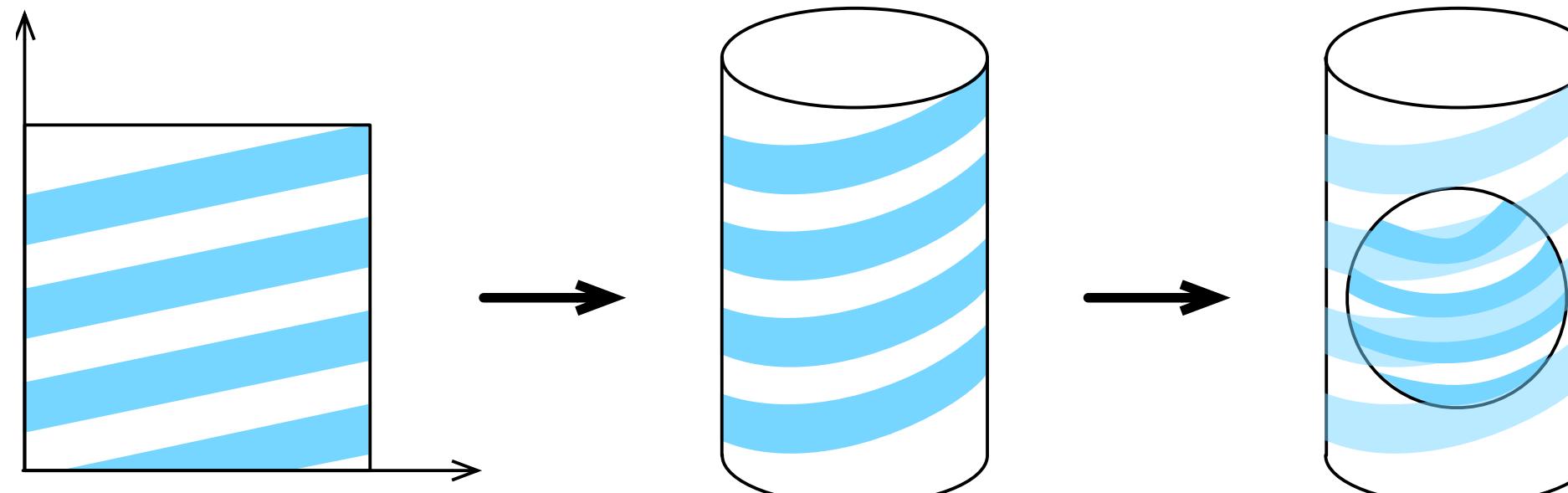
- 纹理 → 中介对象 → 实际对象

- Stage-1: s-mapping, 从纹理到中介对象

- $(u, v) \rightarrow S(x, y, z)$
 - 常见中介包括：平面，长方体，圆柱体，球体，等

- Stage-2: o-mapping, 从中介对象到实际对象

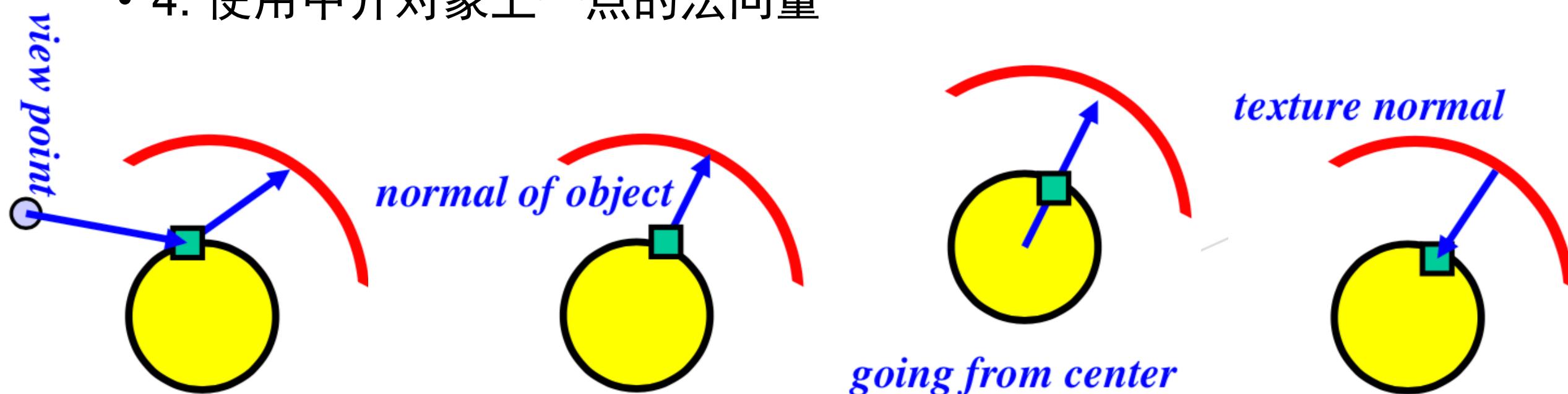
- $S(x, y, z) \rightarrow O(x, y, z)$



- O-mapping

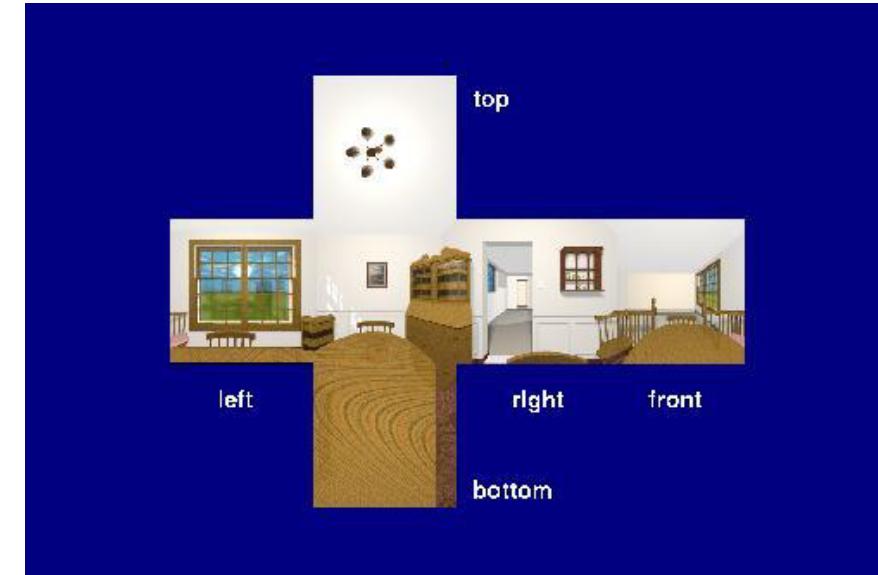
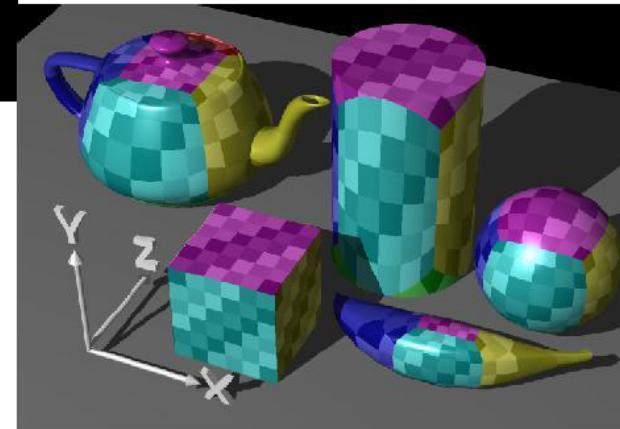
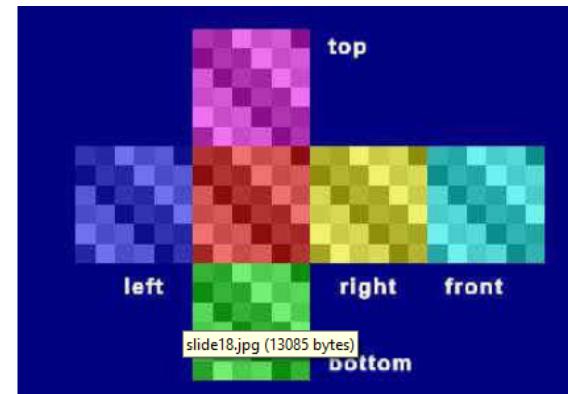
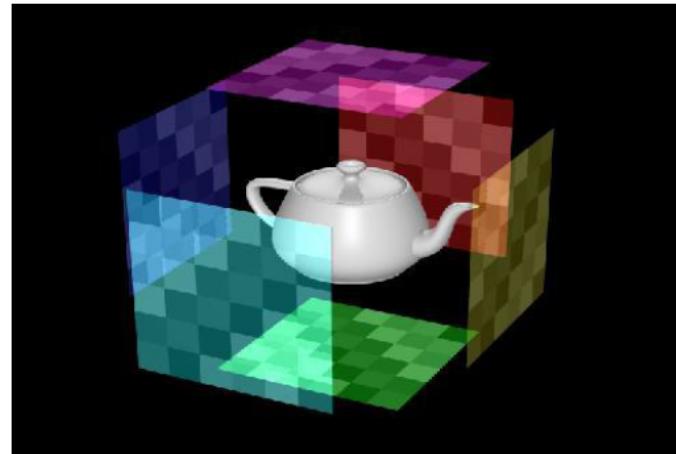
- 计算中介对象与实际对象上点的映射关系

- 该映射关系并非唯一的
 - 1. 使用实际对象上一点反射的视线（依赖于视角）
 - 2. 使用实际对象上一点的法向量
 - 3. 使用实际对象上从中心指向一点的向量
 - 4. 使用中介对象上一点的法向量



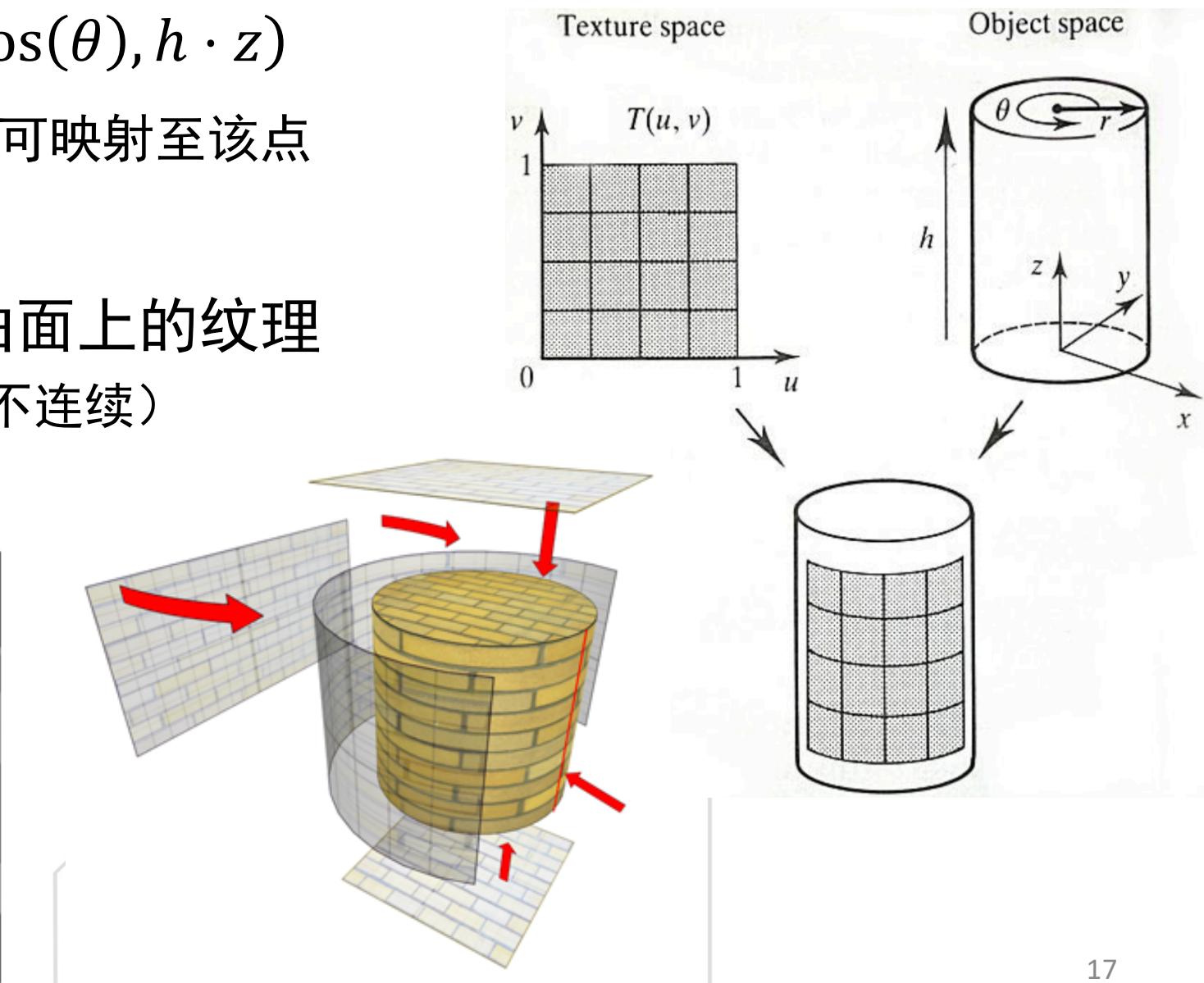
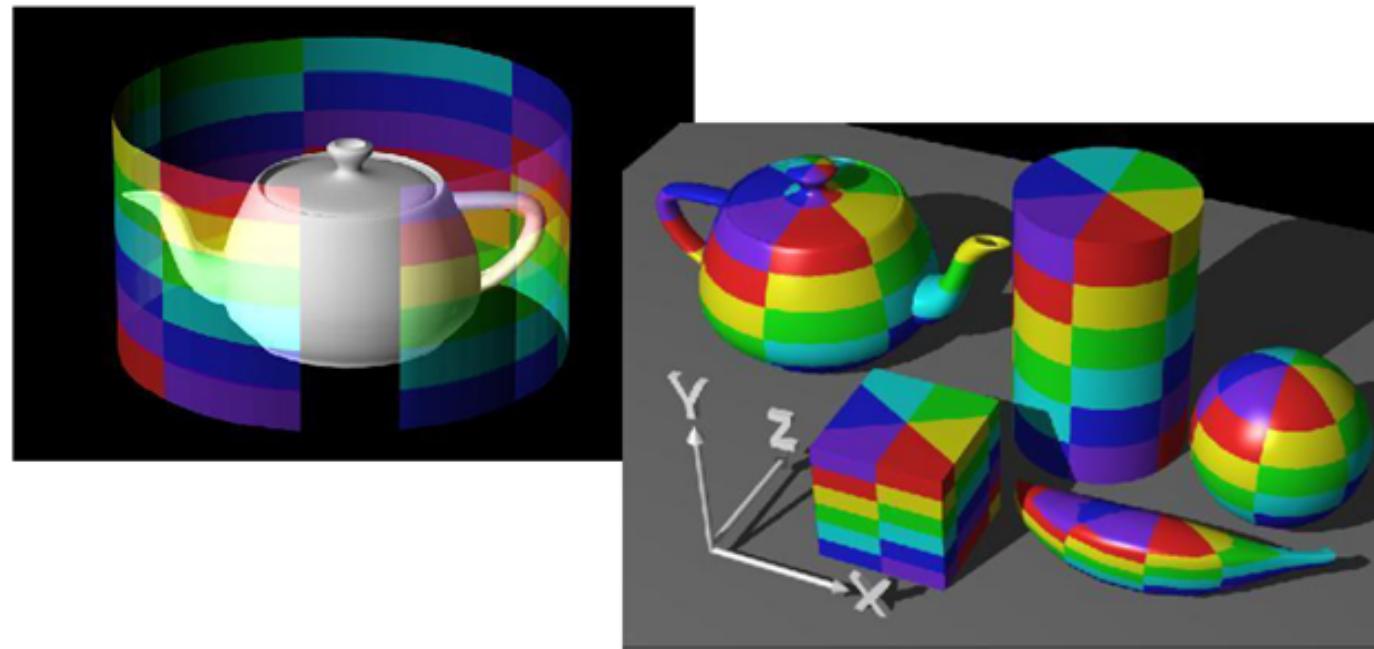
• Cube mapping

- 易于计算：使用正交投影
- 常用于环境映射



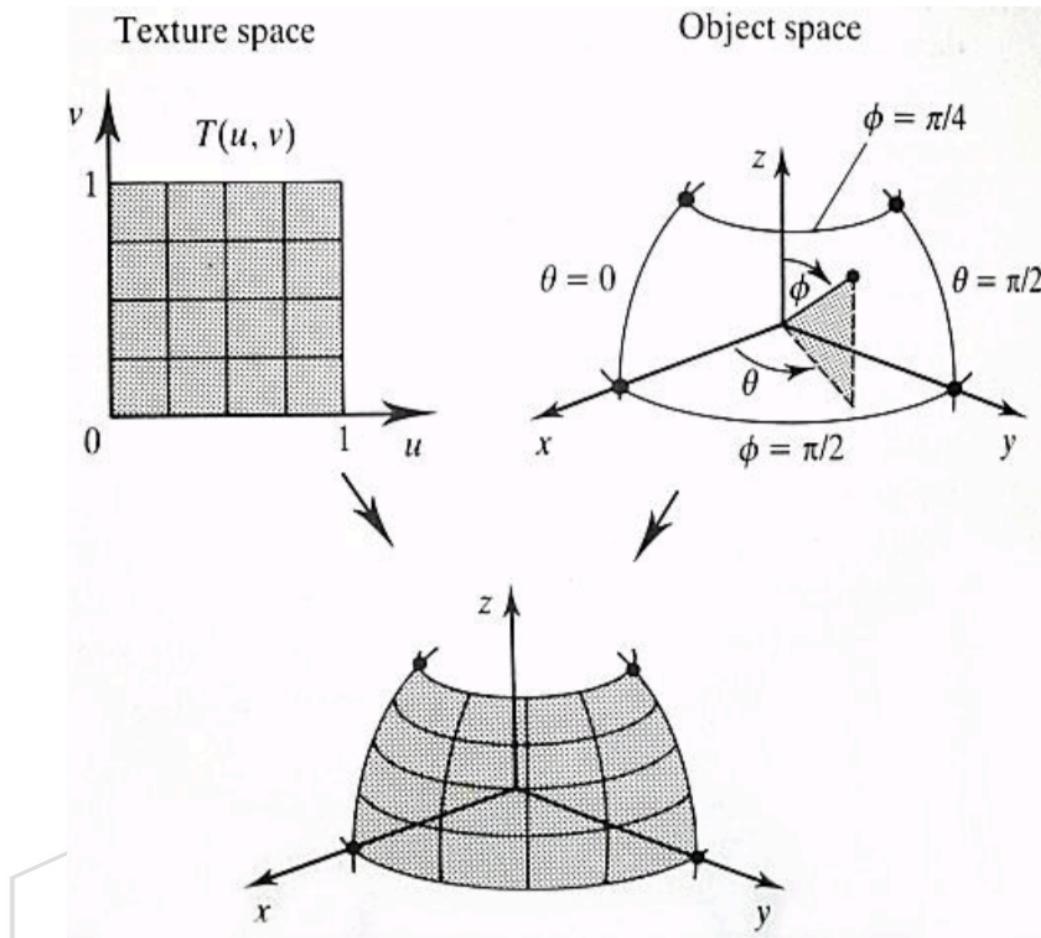
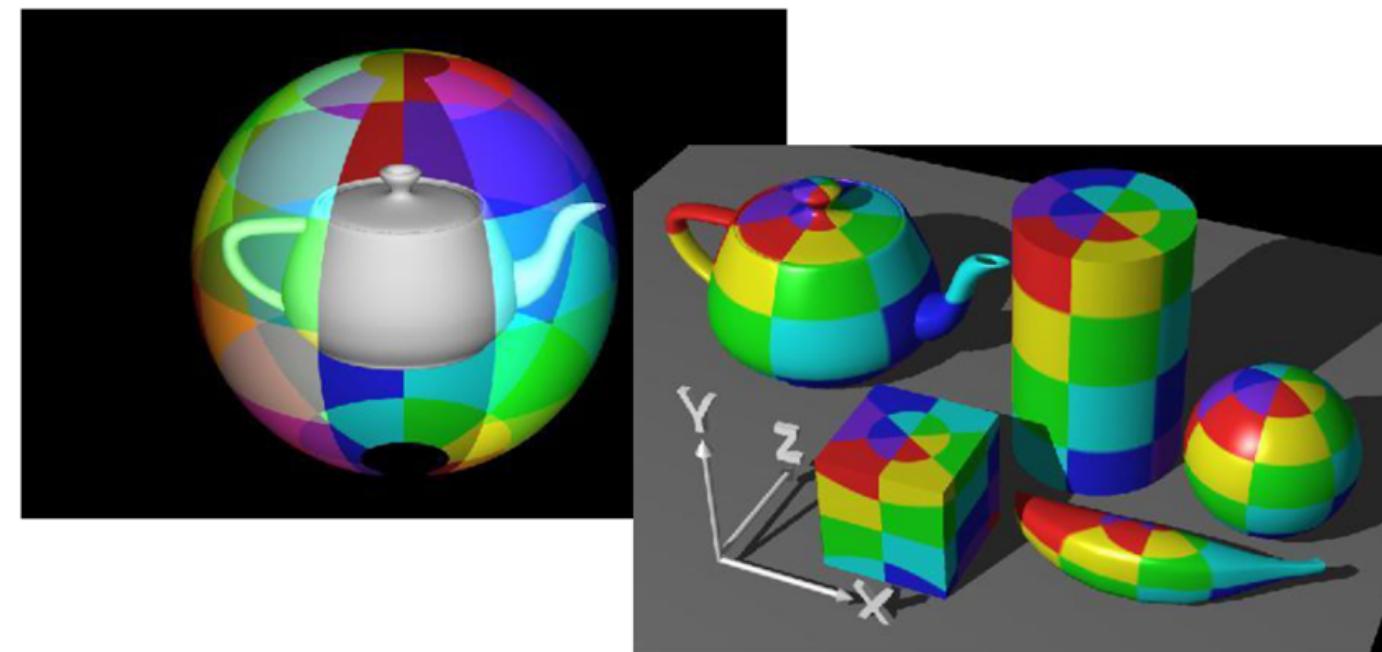
- Cylindrical mapping

- 假设纹理坐标在单位正方形 $[0,1]^2$ 内变化，圆柱高 h ，半径为 r
- 圆柱上一点可表示为 $(r \cdot \cos(\theta), r \cdot \sin(\theta), h \cdot z)$
 - 2D纹理图像上任意一点 $(u, v) = (\frac{\theta}{2\pi}, z)$ 可映射至该点
- 从纹理坐标到圆柱上没有变形
- 适合于与无底的圆柱面拓扑同构的曲面上的纹理
 - 否则需要额外的顶部及底部（可能造成不连续）



• Spherical Mapping

- 球面上的一点可表示为 $(r \cdot \cos \theta \sin \theta, r \cdot \sin \theta \sin \theta, r \cdot \cos \phi)$
- 类似于地图中的映射
 - 无法避免变形，在靠近极点处失真尤其严重
 - 近年来常使用保角映射代替上述参数化映射
- 可用于环境映射中

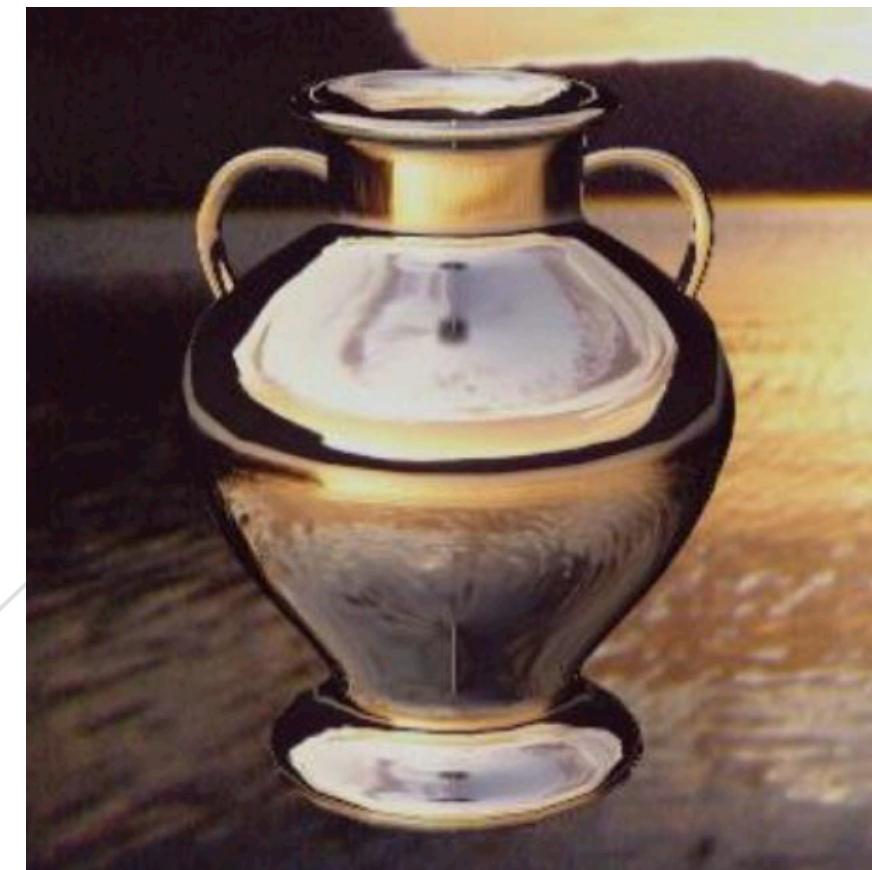
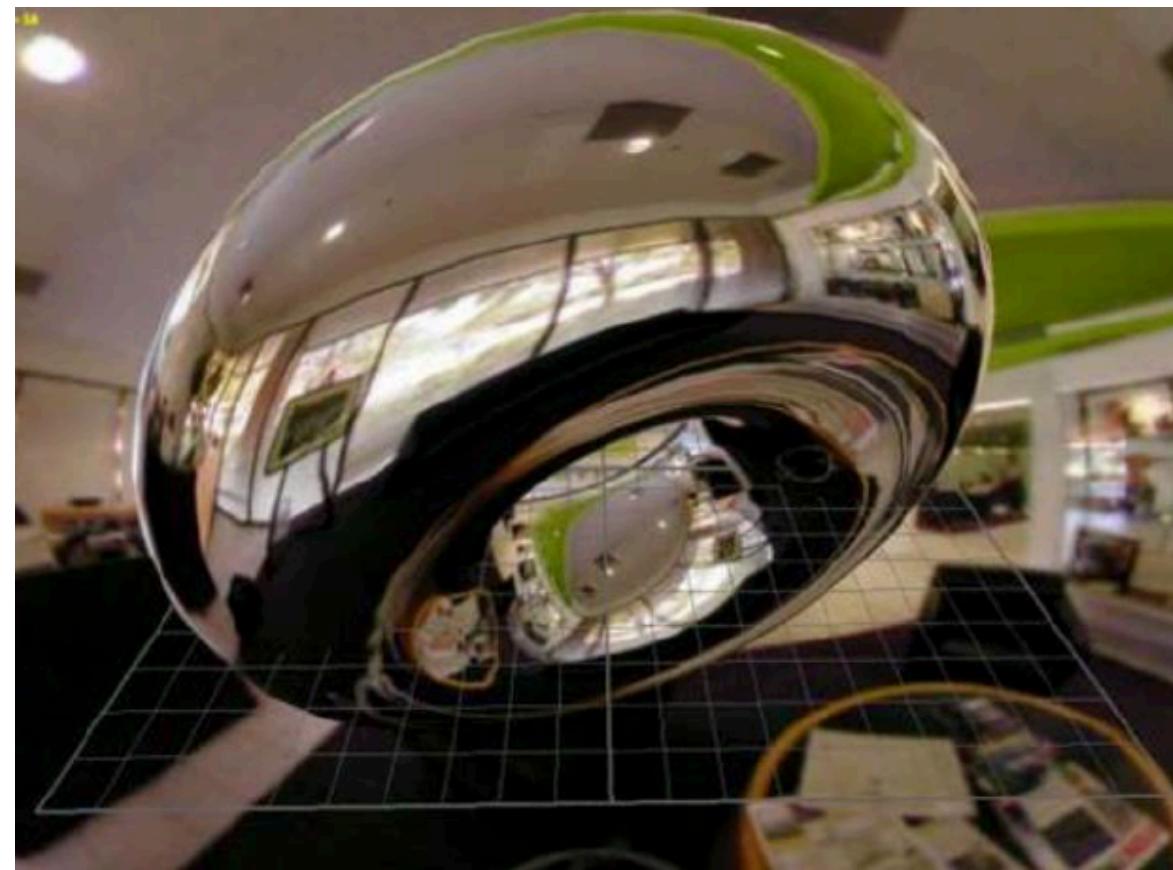


- Planar mapping vs cylindrical mapping



- 环境贴图/反射贴图 (environment/reflection mapping)

- 使用环境图片填充多边形近似光滑表面对环境的反射

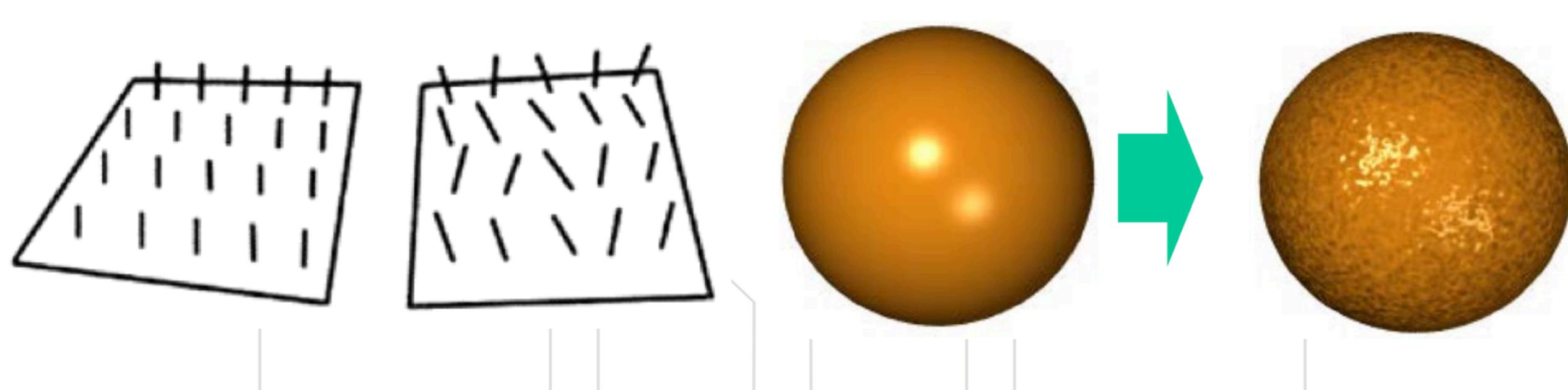


- 纹理简介
- 纹理贴图方法
- 凸凹贴图与移位贴图
- OpenGL中使用纹理



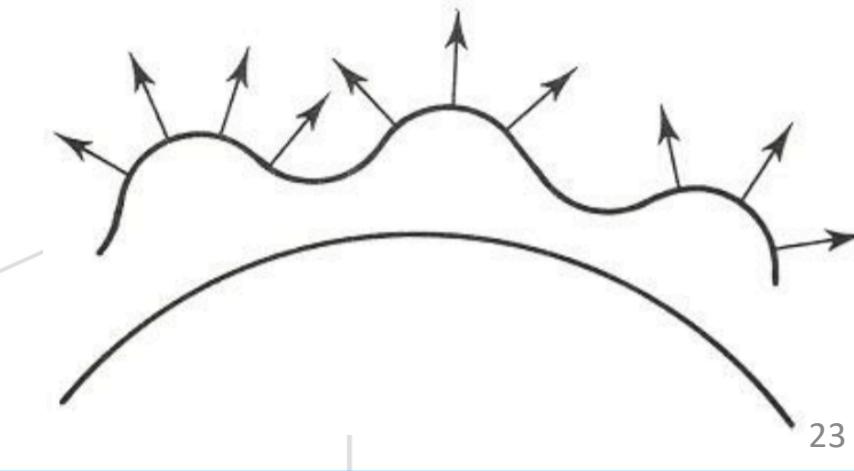
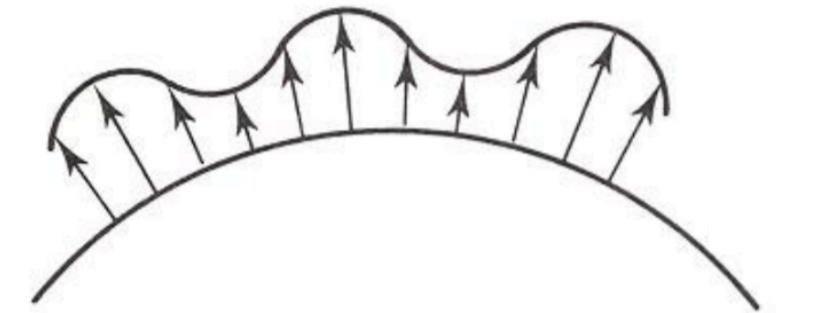
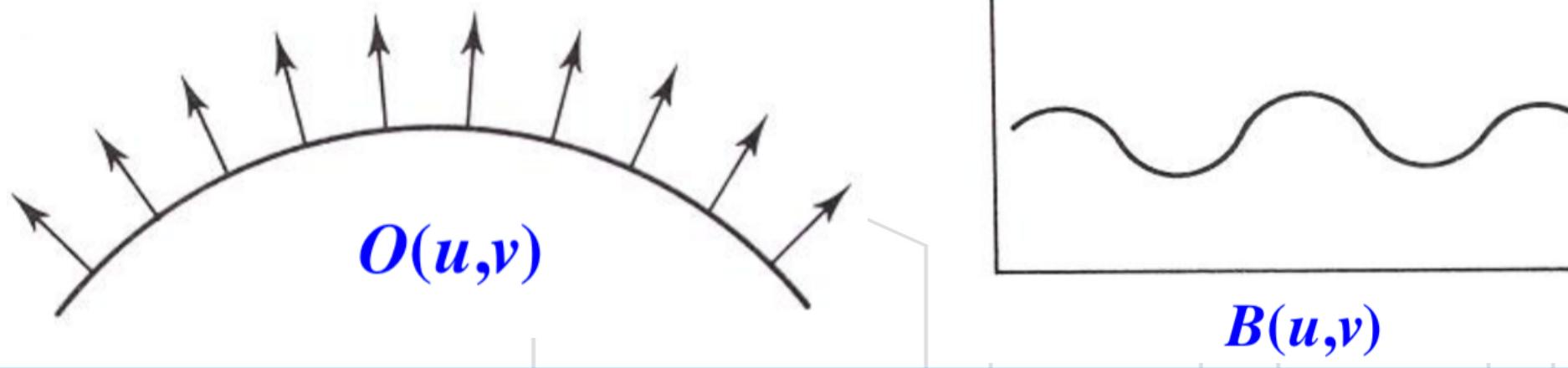
● 凹凸贴图 (bump mapping)

- 在三维环境中通过纹理方法模拟表面凹凸不平的视觉效果
 - 如皱褶，波浪等
- 改变表面光照方程中的**法向量**，而不是表面的**几何形状**本身
 - 光照影响我们对物体三维形状的感知，而法向量改变光照效果
 - 对表面上每个点的法向量增加一个小小的**扰动**，从而给几何形状增加波纹



● 凹凸贴图 (bump mapping)

- 增加扰动的方式并非唯一
 - 在此，我们介绍一种基于参数化曲面的方法
- 原物体表面为 $O(u, v)$ 而凹凸贴图表示为 $B(u, v)$
- 将原表面根据凹凸贴图沿法向量方向移动
 - 从而产生不光滑（凹凸）的表面
- 实际计算中，只更新法向量而不改变位置



● 凹凸贴图 (bump mapping)

– 令 $p = (u, v)$ 为参数化曲面上的一点，曲面在该点的偏微分为

$$\bullet p_u = \left[\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u} \right], p_v = \left[\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v} \right]$$

– 该点处法向量为 $n = \frac{p_u \times p_v}{|p_u \times p_v|}$

– 该点处根据凹凸贴图 $B(u, v)$ 得到的偏移量为 $d(u, v)$

– 则偏移后位置为 $p' = p + d(u, v)$

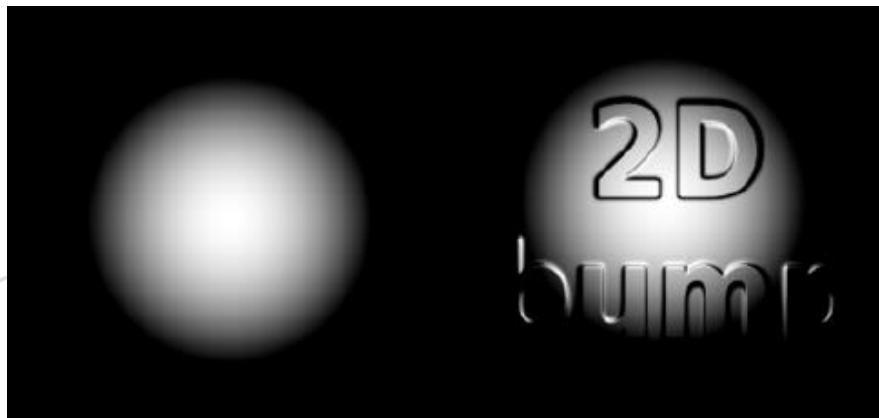
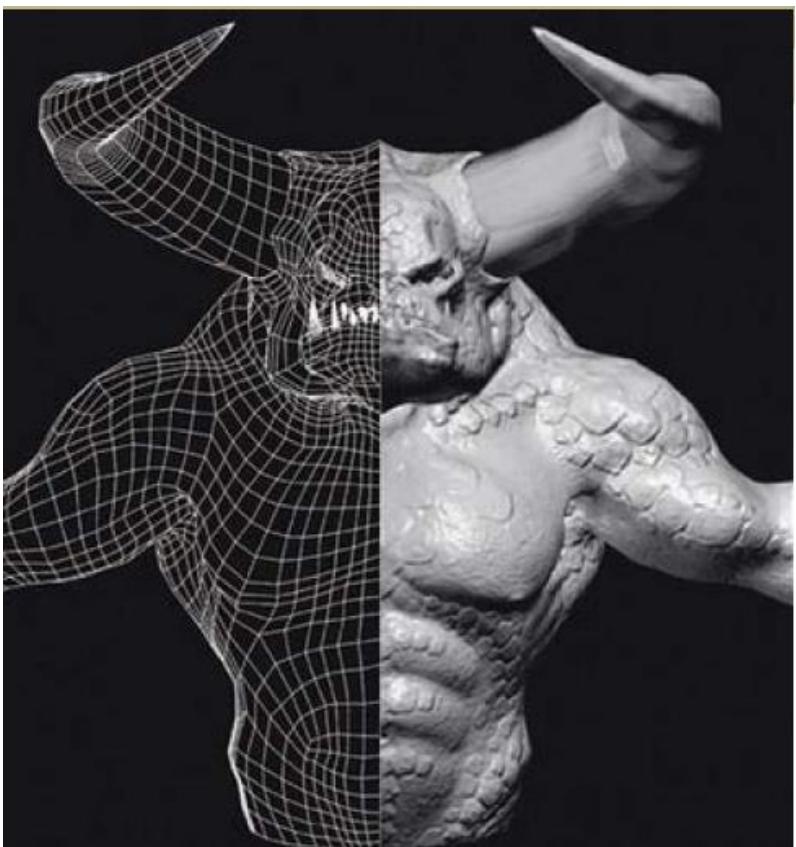
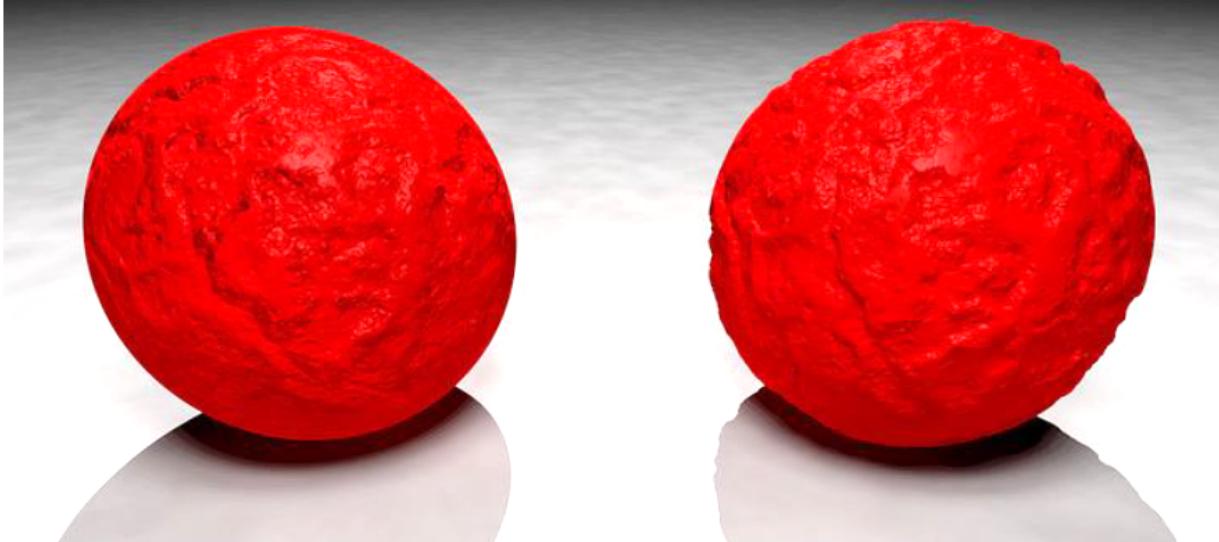
– 需要计算偏移后的法向量 $n' = p'_u \times p'_v$

$$\bullet p'_u = p_u + \frac{\partial d}{\partial u} n + d(u, v) n_u$$

$$\bullet p'_v = p_v + \frac{\partial d}{\partial v} n + d(u, v) n_v$$

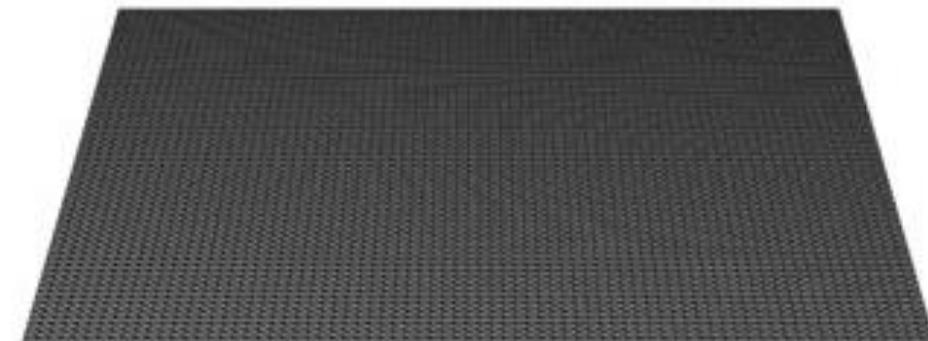


- 凹凸贴图 (bump mapping)



● 移位贴图 (displacement mapping)

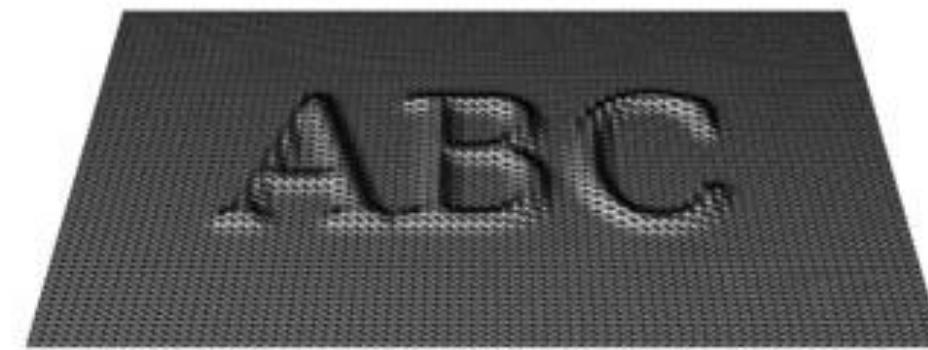
- 移位贴图是凹凸贴图的一种取代方案
- 对点的几何位置进行实际的移位
 - 同样，让点沿法向量方向移动
- 使贴图具备表面纹理的细节和深度表现能力
- 允许自我遮盖，自我投影和呈现边缘轮廓
 - 然而需要大量额外的几何结构支持
 - 此类方法中开销最高的



ORIGINAL MESH



DISPLACEMENT MAP



MESH WITH DISPLACEMENT

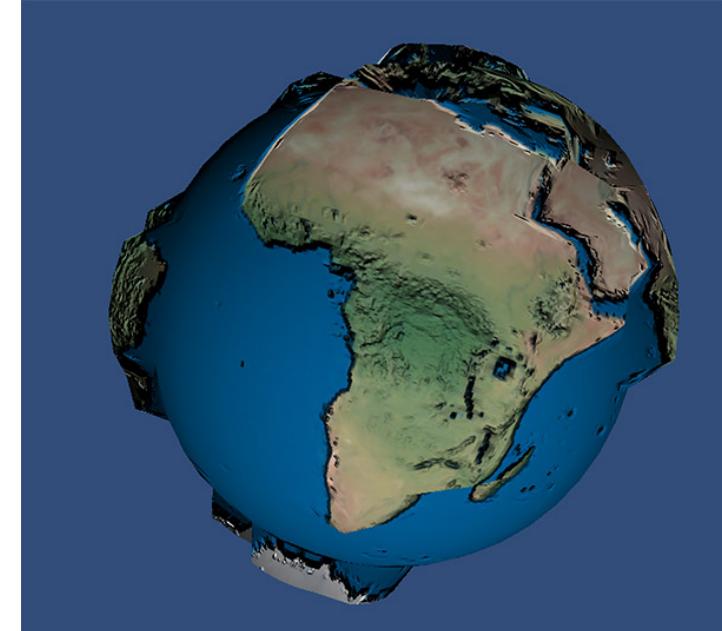


- 移位贴图 (displacement mapping)

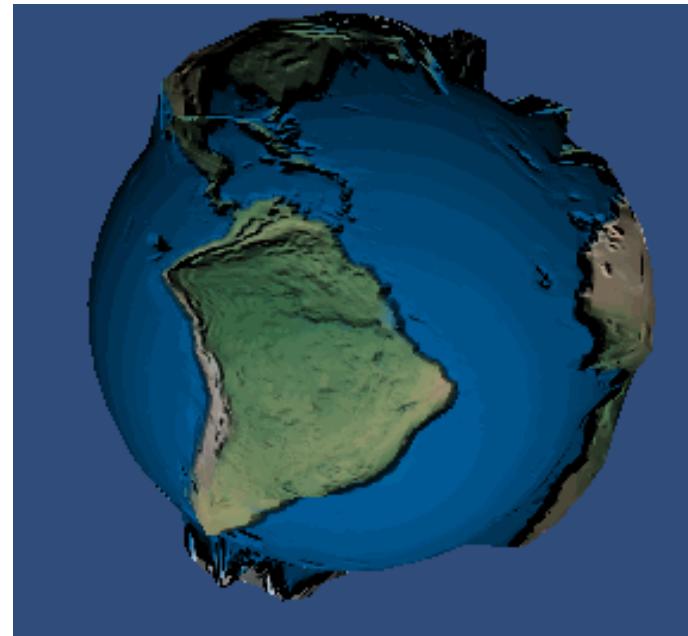
移位前



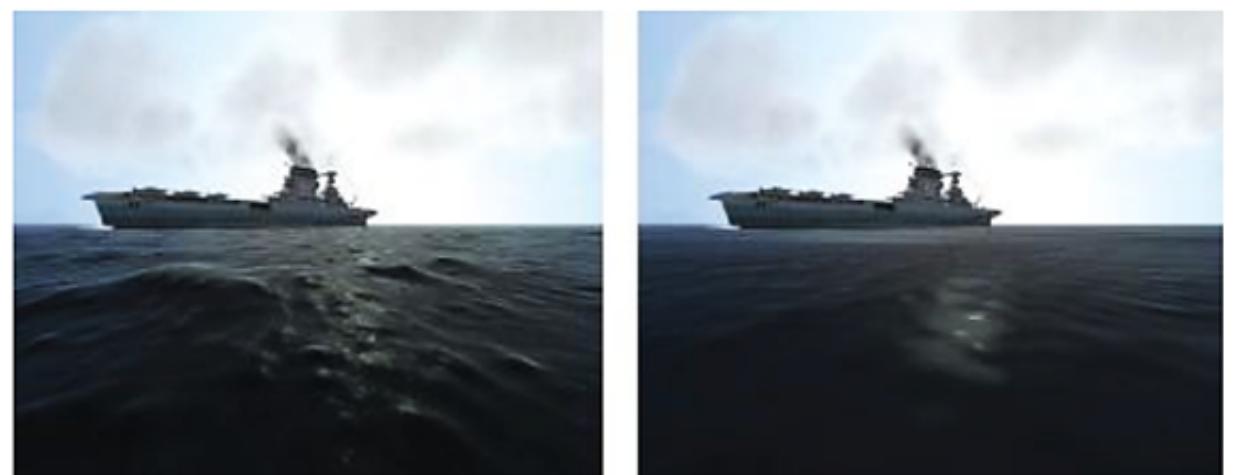
移位后

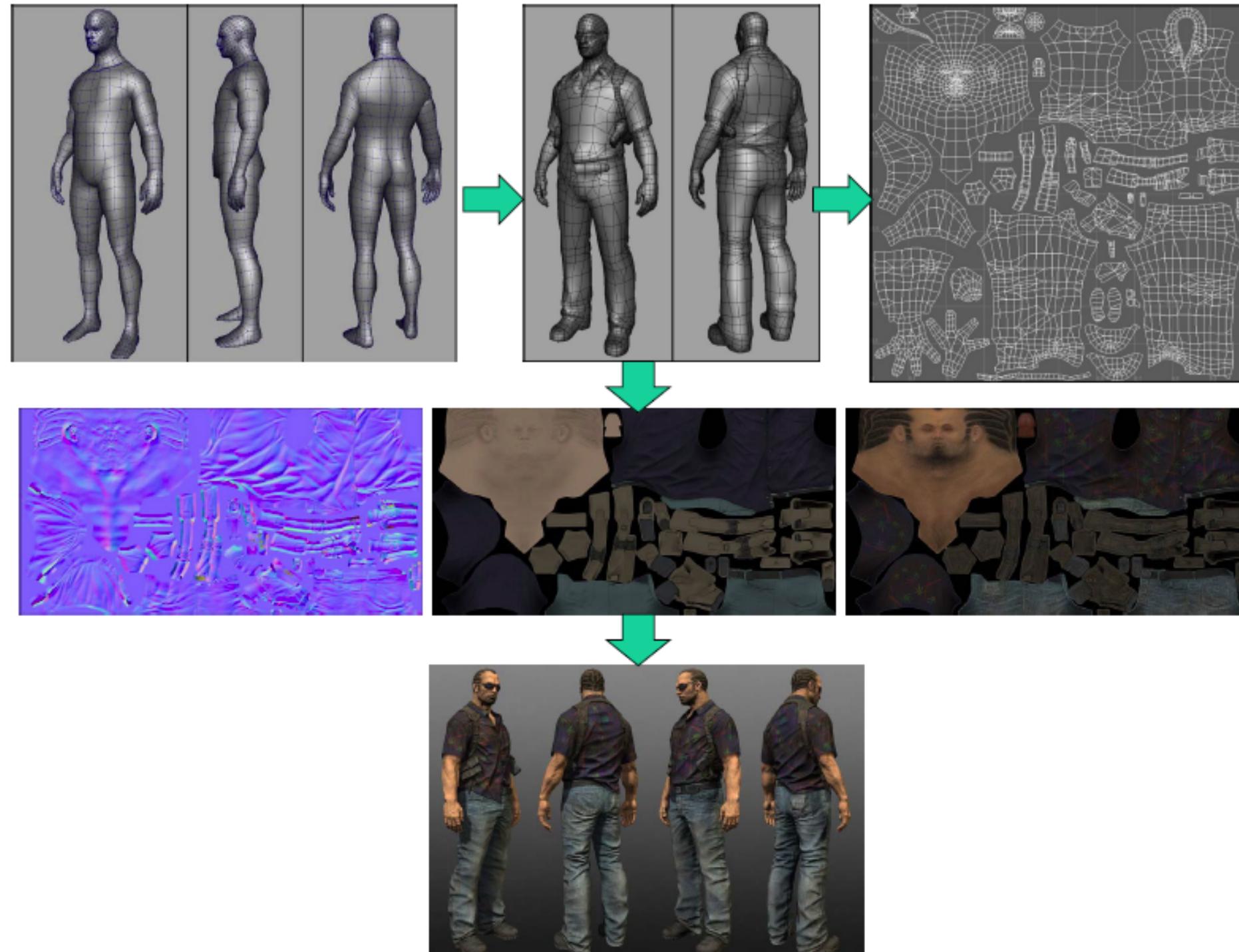


动画



- 移位贴图 (displacement mapping)





- 纹理简介
- 纹理贴图方法
- 凸凹贴图与移位贴图
- OpenGL中使用纹理



● 基本步骤

- 1. 指定纹理
 - 读取或产生纹理图像
 - 将图像赋值给纹理
 - 打开纹理功能
- 2. 指定顶点的纹理坐标
 - 具体的映射函数由应用决定
- 3. 指明纹理参数
 - Wrapping, filtering



● 基本步骤

- 1. 指定纹理
 - 读取或产生纹理图像
 - 将图像赋值给纹理
 - 打开纹理功能
- 2. 指定顶点的纹理坐标
 - 具体的映射函数由应用决定
- 3. 指明纹理参数
 - Wrapping, filtering



● 创建纹理图像

- 使用主内存中的数组指明纹理元素（texels）定义纹理图像
 - `Glubyte my_texels[512][512];`
 - 该数组可由图像文件中读取或使用代码产生
- 打开纹理贴图功能
 - `glEnable(GL_TEXTURE_2D);`
 - OpenGL中支持1-4D纹理
- 创建纹理图像
 - `void glTexImage2D(target, level, components, w, h, border, format, type, *texels)`

● 创建纹理图像

- **void glTexImage2D(target, level, components, w, h, border, format, type, *texels)**

- target: 纹理类型, 如, GL_TEXTURE_2D
- level: mipmap中的级别
- components: 每个纹理元素的分量数
- w, h: 纹理图像的宽与高 (以像素为单位)
- border: “This value must be 0.”
- format, type: 纹理元素的格式与数据类型
- texels: 指向存储纹理图像数组的指针
- 例: **glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0, GL_RGB, GL_UNSIGNED_BYTE, my_texels)**

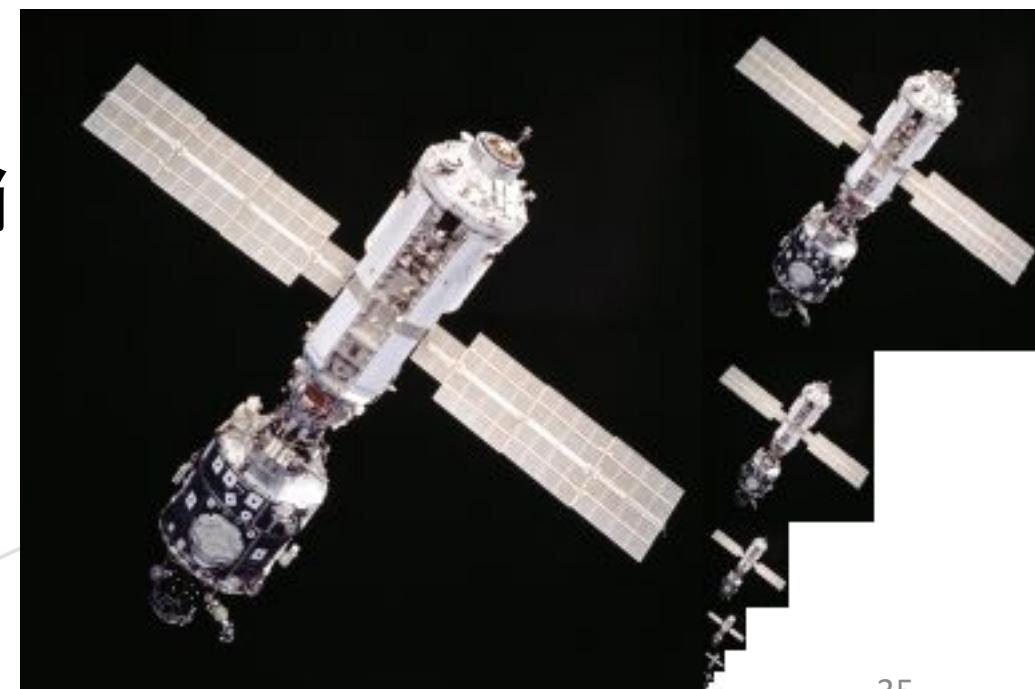
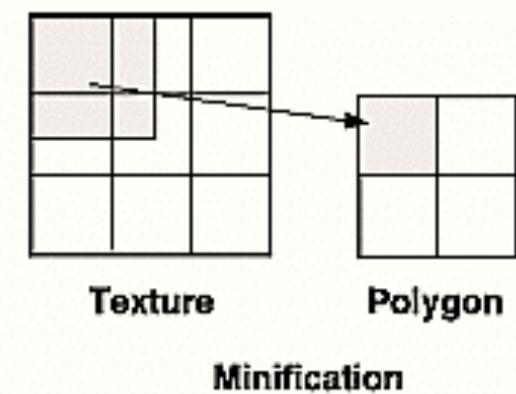
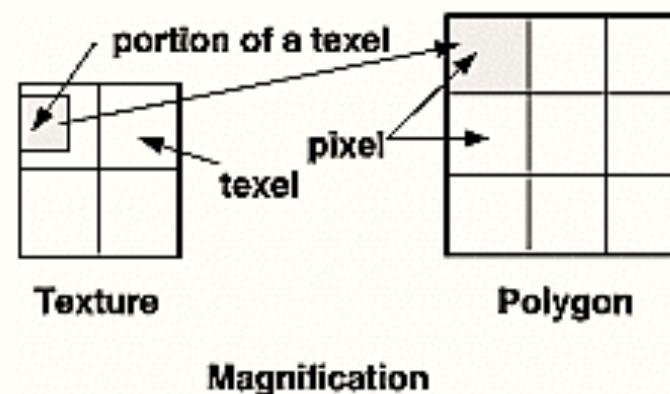
● Mipmap

– MIP (multum in parvo) map

- multum in parvo: 拉丁语, 放置很多东西的小空间
- 1983年, Lance Williams发明

– 为了加快渲染速度、减少图像锯齿, 贴图被处理成由一系列预先计算和优化过的图片组成的文件

- 读取像素远少于普通贴图
- 图片预先进行抗锯齿处理, 减少实时运算开销

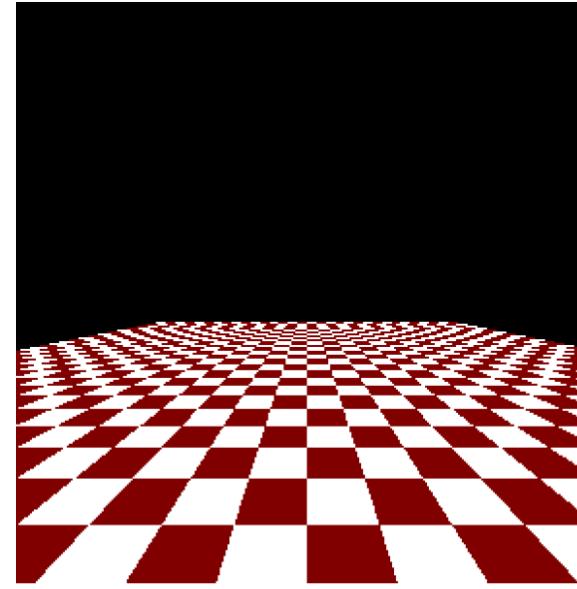


● Mipmap

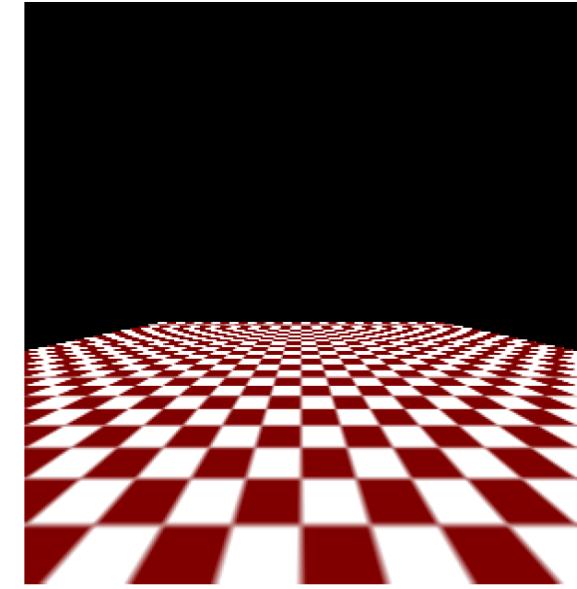
- 可使用**glTexParameterI()**与**glTexImage2D()**提供每个级别下的纹理图像
 - **glTexParameterI(GL_TEXTURE_2D, GL_TEXTURE_BASE_LEVEL, 2);**
 - **glTexImage2D(..., 2, ...);**
- 也可以提供最高分辨率的纹理图像，而使用OpenGL自动创建低级别预处理后的纹理图像
 - **gluBuild2DMipmaps()**
 - **gluBuild2DMipmapsLevels()**
 - 创建指定的级别子集

● Filtering

- OpenGL提供多种简单高效的filtering
 - 通过**glParameteri()**指明
 - 对放大与缩小可使用不同filtering方式
 - GL_TEXTURE_MAG_FILTER
 - GL_TEXTURE_MIN_FILTER
 - Filtering方式
 - GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_LINEAR
- 如, **glParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILLTER, GL_NEAREST)**



GL_NEAREST



GL_LINEAR

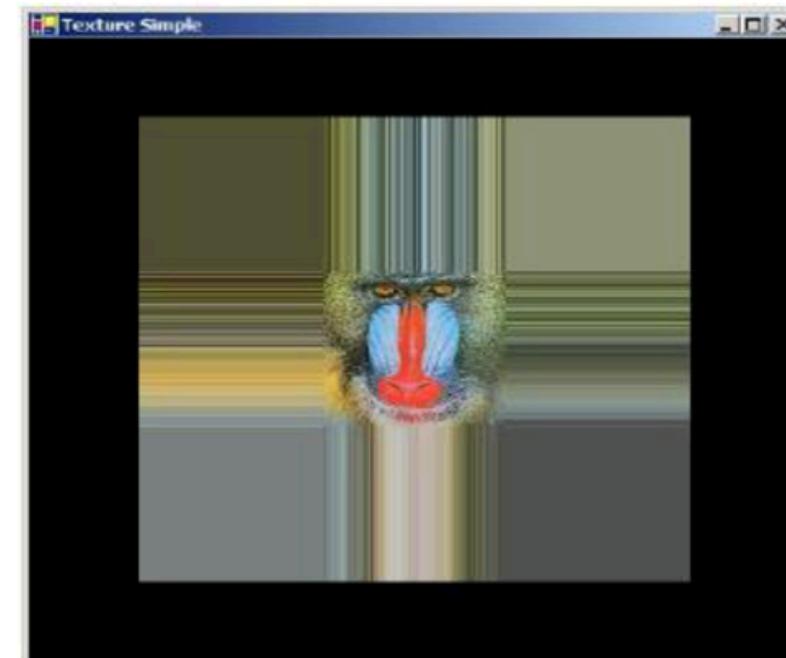


● Wrapping

- 表明如何处理超出纹理坐标边界的点
 - 纹理坐标在(0,0)到 (1,1)的范围内
- 使用 `glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, ...)`
 - `GL_REPEAT` 与 `GL_CLAMP`



GL_REPEAT



GL_CLAMP

Questions?

