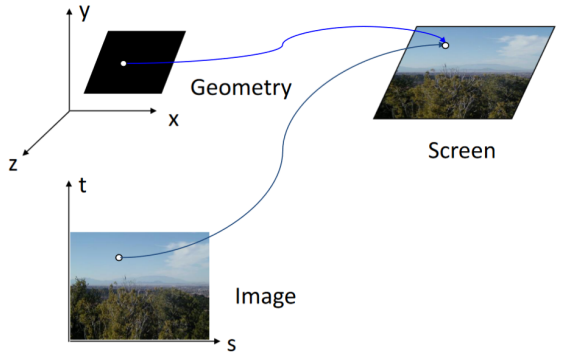
CPT205W10_纹理映射

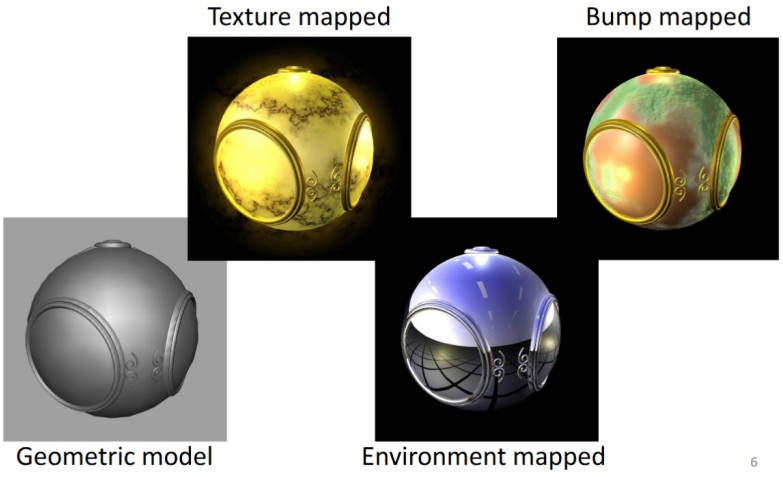
概念

向模型添加细节不一定需要增加模型面数，可以使用贴图来节约开销。



纹理映射的棘手之处在于，矩形纹理可以映射到非矩形区域，这必须以合理的方式完成。映射纹理时，由于应用了各种转换（旋转、平移、缩放和投影），它在屏幕上的显示可能会失真。

类型：纹理映射，环境映射（反射映射 environment/ reflection mapping），凹凸映射（Bump mapping）



指定纹理

两种类型：

- image：使用2D图像作为纹理
- 程序：使用程序生成纹理

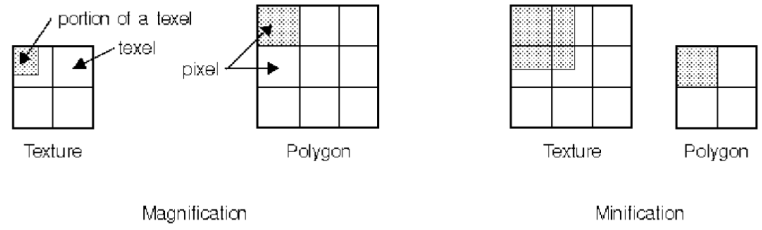
纹理可定义为1，2，3维图案

纹理数组中的各个值通常称为纹素（**texels** (纹理元素 texture elements).）。指向包含纹理图像数据的纹素数组的指针。此数据描述纹理图像本身及其边框。

纹理图案中的每个颜色定义可以包含四个 RGBA 分量（png），每个分量的意义你可以自定义

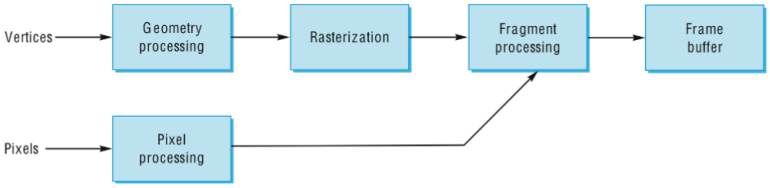
放大和缩小（Magnification and minification）

根据纹理大小、扭曲和屏幕图像的大小，一个纹素可能映射到多个像素（称为放大），一个像素可能被多个纹素覆盖（称为缩小）。

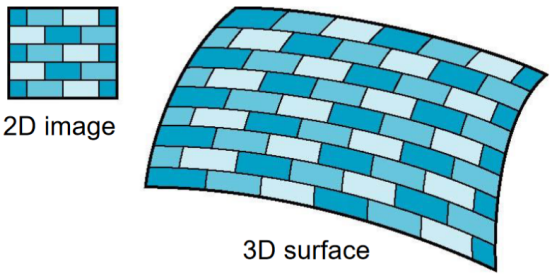


由于纹理由离散的纹素组成，因此必须执行筛选操作以将纹素映射到像素。

如果许多纹素对应于一个像素，则会将它们平均化以适合;如果纹素边界跨越像素边界，则执行适用纹素的加权平均值。

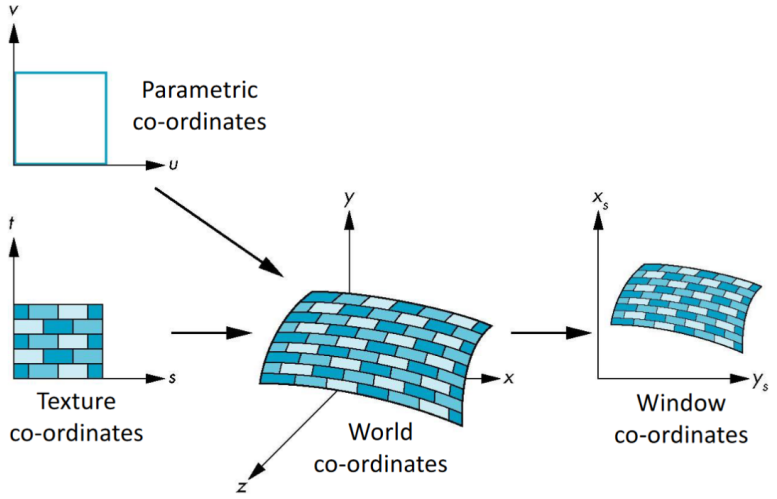


纹理映射在渲染管道的末尾实现（已经clipper后）。将图像映射到图形表面涉及3-4个坐标系，并不简单。



用于纹理映射的坐标系：

- 参数坐标（Parametric co-ordinates）：用于对曲线和曲面进行建模
- 纹理坐标（Texture co-ordinates）：用于标识图像中要映射的点
- 对象或世界坐标（Object or world co-ordinates）：映射发生的地方
- 窗口坐标（window co-ordinates）：最终生成最终图像的位置



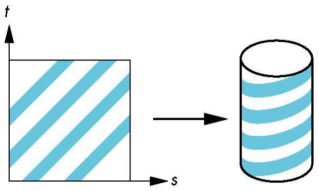
我们逆向思考，向后映射（backward mapping）即：给定屏幕上一个像素，找到它对应物体对象上的哪个点；给定对象上的一个点，找到它对应纹理中的哪个点。即映射两次。s = s(x,y,z)，t = t(x,y,z)

两步映射（Two-part mapping）

解决后面那个映射的方法，我们可以先将纹理映射到一个简单的中间表面，然后再映射到对象上。

第一步：将纹理映射到中间对象

如圆柱体：



A parametric cylinder

x = r*cos(2pi*u)
y = r*sin(2pi*u)
z = h*v

maps a rectangle in the (u,v) space to the cylinder of radius r and height h in the world co-ordinates

s = u
t = v

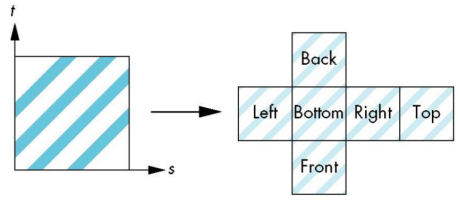
maps from the texture space.

不同的中间表面会导致不同位置的失真。

环境球就是用球体，或者天空盒。

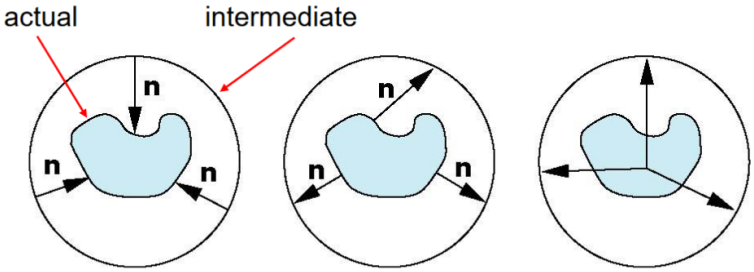
盒体映射（Box mapping）

简单易用的正交投影：



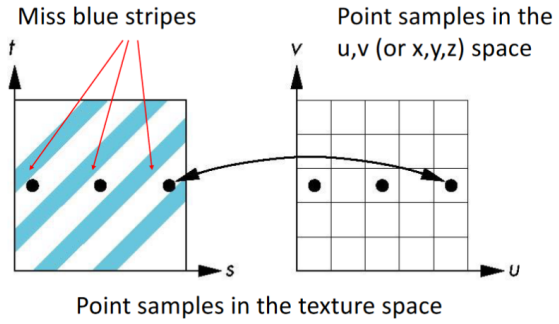
第二步：从中间对象映射到实际对象模型

我们有三种向量：中间对象到模型的法线，模型到中间对象的法线，从中间对象中心的向量

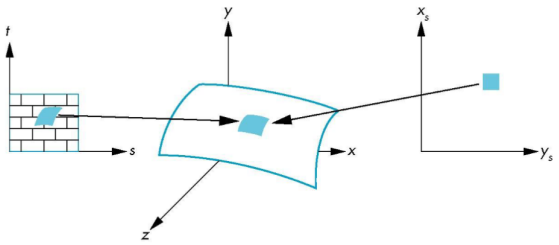


锯齿（Aliasing）

纹理的点采样会导致贴图的锯齿错误：（感觉是欠采样导致的）



一个解决方法是面积平均（Area averaging），但需要付出一些开销：



过程

3个主要步骤：

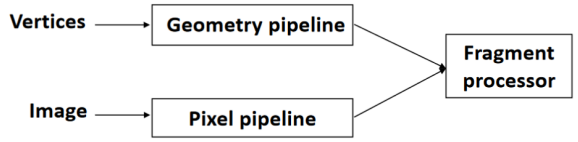
- 指定纹理：读取或生成贴图，分配给纹理，启用纹理
- 将纹理坐标分配给顶点：适当的映射

- 确定贴图参数：mode，filtering，wrapping

glTexImage2D(): Defining image as texture (type, level and colour)
glTexParameter*(): Specifying filtering and wrapping
glTexEnv{fi}[v](): Specifying mode (modulating, blending or decal)
glTexCoord{1234}{sifd}{v}(): Specifying texture co-ordinates s and t while r is set to 0 and q to 1
glTexGen{ifd}[v](): Automatic generation of texture co-ordinates by OpenGL
glHint(): Defining perspective correction hint
glBindTexture(): Binding a named texture to a texturing target

纹理映射和Opengl管线

在像素/片元处理（Fragment Processing）前，图像和模型流经的是不同的管线，互不干扰。



指定纹理 Specifying a texture image

从 CPU 内存中的纹素（纹理元素）数组定义纹理图像`Glubyte my_texels[512][512][4]`

然后将纹理定义为其他的pixcel map

启用纹理映射`glEnable(GL_TEXTURE_2D)`OpenGL 支持 1-4 维纹理映射

如：

- The texture (OpenGL logo) is a 256 x 256 image.
- It is mapped to a rectangular polygon.
- The polygon is viewed in perspective.

将图像定义为纹理：

```
glTexImage2D(target, level, components, w, h, border, format, type, texels);
```

- target: 纹理类型，如GL_TEXTURE_2D
- level：用于 mipmapping

- components: 每个纹素的颜色元素 - 一个从 1 到 4 的整数，表示选择哪些 RGB 和 A 组件进行调制或混合。值 1 选择 R 分量，2 选择 R 和 A 分量，3 选择 RGB，4 选择 RGB 和 A。
- w和h：图像的宽度和高度（以像素为单位）

  w 和 h 都必须具有 $2^m + 2b$ 形式，其中 m 是整数，b 是 border 的值，尽管它们可以具有不同的值。纹理贴图的最大大小取决于 OpenGL 的实现，但必须至少为 64 × 64（或带边框的 66 × 66）。
- border：用于平滑（稍后讨论），通常为 0。
- format 和 type：描述纹理图像数据的格式和数据类型，与`glDrawPixels()`有相同含义。

  format参数：GL_COLOR_INDEX, GL_RGB, GL_RGBA, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_LUMINANCE, GL_LUMINANCE_ALP，除了GL_STENCIL_INDEX 和 GL_DEPTH_COMPONE之外与`glDrawPixels()`相同

  type参数：GL_BYTE, GL_UNSIGNED_BYTE, GL_SHORT, GL_UNSIGNED_SHORT, GL_INT, GL_UNSIGNED_INT, GL_FLOAT, GL_BITMAP

如：`glTexImage2D(GL_TEXTURE_2D, 0, 3, 514, 514, 1, GL_RGB, GL_UNSIGNED_BYTE, my_texels);`

纹理参数

Opengl有多种参数来确定纹理的应用方式：

- 滤波器模式（Filter modes）允许使用面积平均而不是点采样。
- 缠绕参数（Wrapping parameters）可以确定若s和t超过[0,1]范围后会发生什么
- 模式/环境参数（Mode/environment parameters）决定纹理映射如何与着色交互。
- Mip贴图（Mipmapping）使我们可以用多种分辨率的纹理。

滤波控制 Controlling filtering

OpenGL 允许指定任意滤波选项来确定计算方式。这些选项在速度和图像质量之间提供了不同的权衡。我们可以独立指定放大和缩小的过滤方法。但Opengl一般可以提供最佳效果。

使用 `glTexParameter*()` 指定放大(magnification)和缩小(minification)滤波方法：

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEA);
```

注意，线性筛选需要一个额外纹素的边框，以便在边缘进行筛选 （border = 1）。

纹理缠绕 Texture wrapping

可以分配范围 [0,1] 之外的纹理坐标，并让它们在纹理映射中固定或重复。

一种可能（Wrapping）是假设有一个大平面，纹理坐标在两个方向上从 0.0 到 10.0，我们将在屏幕上获得 100 个纹理副本（10x10）。在重复过程中，纹理坐标的整数部分将被忽略，并且纹理贴图的副本会平铺表面。`glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)`

另一种可能性（Clamping）是固定纹理坐标：任何大于 1.0 的值都设置为 1.0，任何小于 0.0 的值都设置为 0.0。`glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)`

纹理模式 Texture mode

控制纹理的应用方式`glTexEnv{fi}[v](GL_TEXTURE_ENV, prop, param)`

GL_TEXTURE_ENV_MODE

- `GL_REPLACE`仅使用纹理颜色
- `GL_MODULATE`使用计算的阴影进行调制
- `GL_BLEND`与环境色混合

使用GL_TEXTURE_ENV_COLOR设置混合色

可以从三个参数中选择一个，提供给`glTexEnv{if}{v}(GLenum target, GLenum pname, TYPEparam)`

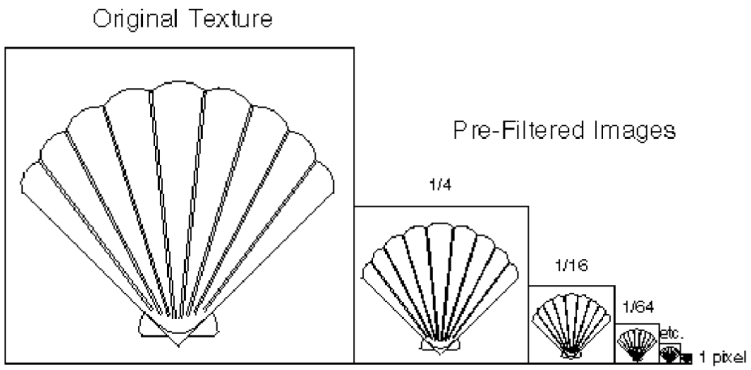
若只使用纹理颜色作为最终颜色，称为贴花模式（decal mode），纹理绘制在片段顶部。使用纹理来调制或缩放片段颜色，这对于将照明效果与纹理相结合非常有用。

可以根据 Texture 值将 Constant 颜色与 Fragment 颜色混合。

- The **target** must be **GL_TEXTURE_ENV**. If **pname** is **GL_TEXTURE_ENV_MODE**, **TYPEparam** can be **GL_DECAL**, **GL_MODULATE**, or **GL_BLEND**, to specify how texture values are to be combined with the colour values of the fragment being processed. In the decal mode and with a three-component texture, the texture colours replace the fragment colours.
- With either of the other two modes or with a four-component texture, the final colour is a combination of the texture and the fragment values. If **pname** is **GL_TEXTURE_ENV_COLOR**, **TYPEparam** is an array of four floating-point values representing the R, G, B and A components. These values are used only if the **GL_BLEND** texture function is specified as well.

Mip贴图 Mipmapping

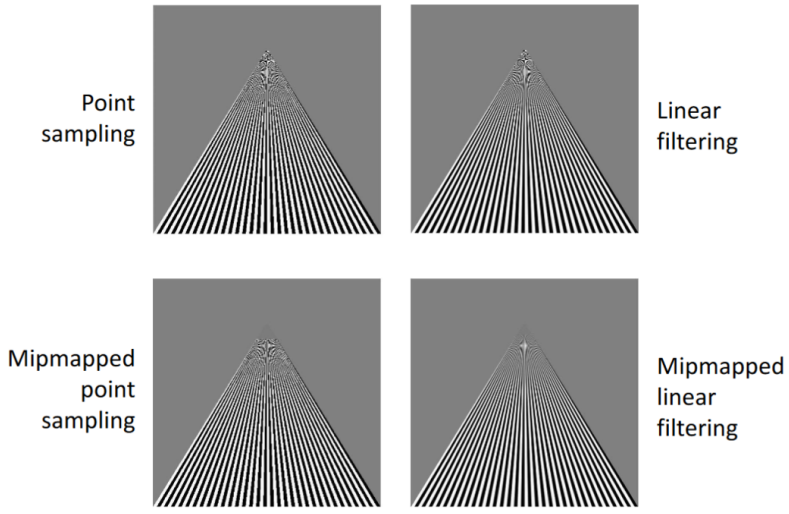
随着纹理对象远离视点，纹理贴图的大小必须随着投影图像的大小而减小。为了实现这一点，OpenGL 必须将纹理映射过滤到合适的大小，以便映射到对象上，而不会引入视觉上令人不安的伪影。



Mipmapping 需要一些额外的计算来减少插值误差，但是，如果不使用 Mipmap，映射到较小对象的纹理可能会随着对象的移动而闪烁。（摩尔纹）

为了使用 mipmapping，我们以 2 的幂次方提供所有大小的纹理，介于最大大小和 1×1 映射之间。例如，如果最高分辨率的地图是 64×16，我们还必须提供大小为 32×8、16×4、8×2、4×1、2×1 和 1×1 的map。

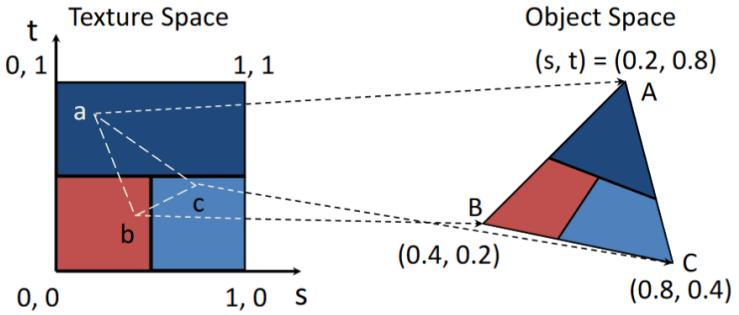
通过调用`glTexImage2D(GL_TEXTURE_*D, level, …)`给每种分辨率1一次



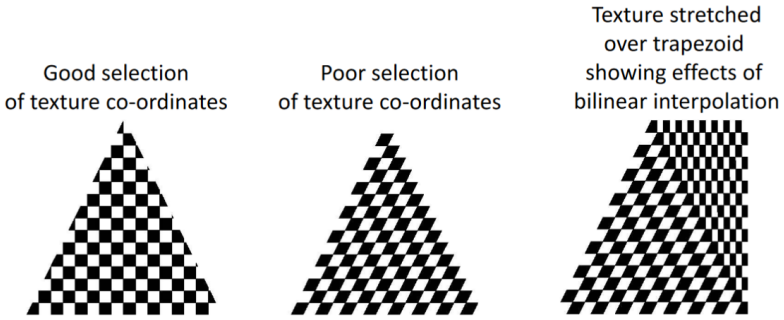
生成纹理坐标（texture co-ordinates）

我们需要指示纹理应如何相对于要应用的片段对齐。我们可以在指定场景中的对象时指定纹理坐标和几何坐标。

指定纹理坐标的命令`glTexCoord*()`，使用方式类似于glColor*()，在glBegin()和glEnd()间使用。



OpenGL 使用插值从指定的纹理坐标中查找合适的纹素。可能会有扭曲。



OpenGL 可以自动生成纹理坐标`glTexGen{ifd}[v](GLenum coord, GLenum pname, GLint param)`

指定一个平面并根据与平面的距离生成纹理坐标

coord坐标：GL_S, GL_T, GL_R, GL_Q中的一个

pname生成模式：GL_OBJECT_LINEAR, GL_EYE_LINEAR, GL_SPHERE_MAP (用于环境贴图)其一

透视矫正提示 Perspective correction hint

纹理坐标和颜色插值：在屏幕空间中线性，或使用深度/透视值（depth/perspective values）（较慢）

`glHint(GL_PERSPECTIVE_CORRECTION_HINT, hint)`

hint：GL_DONT_CARE, GL_NICEST, GL_FASTES

总结：

- 为什么需要纹理映射，纹理映射的运作原理
- 纹理映射的技术：确定贴图，放大与缩小，两步映射，多级细节（mipmapping），模型，滤波，缠绕。
- 贴图映射的步骤

- OpenGL代码