# 投影纹理映射 Projective Texturing

## 简介

当提到“纹理映射”时，通常会认为是通过明确地分配纹理坐标，将纹理应用到物体表面。然而，这并不是纹理的唯一使用方式。纹理也能够被投影到表面上，如同构成一个幻灯片投影仪，这种技术被称为“投影纹理映射”。



上图是一个纹理投影映射的例子，一个恶魔的纹理被投影到球体、盒子和平面上。需要注意的是，在这个过程中没有对象“阻挡”光线的概念，即纹理投射到球体上后，还能透过球体投影到球体下方的平面上。

## 投影纹理映射的工作原理 How Projective Texturing Works

### 原理简介

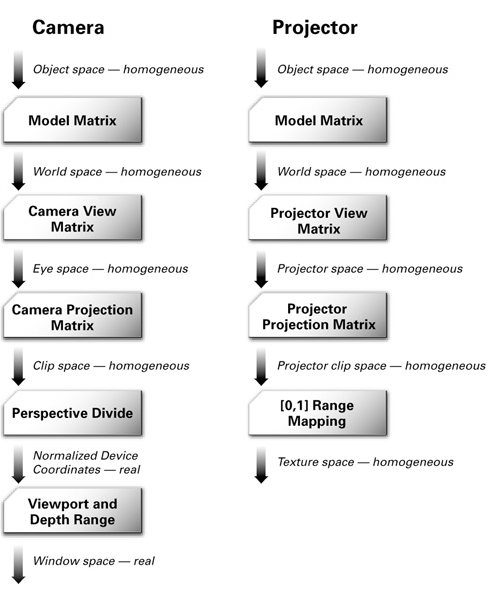
如同常规的纹理映射一样，投影纹理映射需要为场景中的每个三角形找到合适的纹理坐标。因此关键便是如何将正确的坐标分配给各个多边形。

在图形管道中，对象空间中的三角形坐标通过一系列的变换转化成各个坐标系的顶点坐标，再通过投影变换投射到窗口空间，从而使三角形能够被光栅化并被显示在视口中。如果将视口看作一张纹理，这个过程可以看为将三角形的顶点映射到纹理中纹理像素的过程。

投影纹理映射的工作原理与这个过程相同。通过一系列的变换将对象空间的坐标映射到一张2D纹理上，就能找出每个顶点在纹理中的位置，然后将这个位置作为纹理坐标分配给顶点，就可以将贴图正确地分配给每个被渲染的三角形。

### 变换对比 Comparing Transformations

如下图所示，左边的是常规图形管道变换序列，右边是投影纹理映射变换序列。注意在右侧的变换中，坐标始终保持在齐次空间Homogeneous Space。



### 齐次纹理坐标 Homogeneous Texture Coordinates

纹理坐标通常是二维的(s, t)，也存在三维纹理坐标(s, t, r)。如同齐次位置坐标有第四成分w一样，齐次纹理坐标也有第四个成分，称为q。通过它我们可以在纹理空间中使用4x4矩阵表达投影、旋转、平移和缩放变换。

执行投影纹理查找时，硬件会在使用(s, t, r, q)除以q，得到(s/q, t/q, r/q, 1)。与顶点着色器程序运行后执行的透视划分类似，坐标将会被正确地对应到一个2D或3D纹理中。

## 投影纹理映射的实现 Implementing Projective Texturing

### 在顶点着色器中计算纹理坐标

计算投影后纹理坐标的公式如下：

0.5, 0.0, 0.0, 0.0

(s, t, r, q)=(x0, y0, z0, w0)·ModelMatrix·ViewMatrix·ProjectionMatrix·( 0.0, 0.5, 0.0, 0.0)

0.0, 0.0, 0.5, 0.0

0.5, 0.5, 0.5, 1.0

* (x0, y0, z0, w0)是模型顶点坐标
* ModelMatrix是模型变换矩阵
* ViewMatrix是视图矩阵
* ProjectionMatrix是投影矩阵
* 乘以投影矩阵后，s，t，r，q在[-1, 1]之间取值，因此公式最后需要一个额外的矩阵用于保持s，t，r，q在[0, 1]之间取值
* 我们把公式中所有矩阵相乘得到的变换矩阵称为纹理矩阵Texture Matrix

### 需要注意的问题 Notable Issues

#### 通过上述方式实现的投影纹理映射由于没有执行遮挡检查，导致纹理会穿透物体，映射到视锥内的每一个三角形上。

#### 当q坐标为负数时，会导致反投影的副作用，在老式GPU中可能会导致不确定的结果。

有几种方法可以用来避免q为负数带来的副作用：

* 只绘制在光源前方的几何体
* 移除在光源后方的几何体
* 将反投影因子加入到3D衰减纹理Attenuation Texture
* 在像素着色器中检查q是否为负值（常用手段）