CGBOOK

**环境映射**

环境映射的目的是模拟光滑物体表面对周围环境的反射的技术，例如反射厨房的闪亮水壶或反射街道的精美抛光车。 由于通过对象的反射可以看出环境，所以被称为映射到物体上。 Blinn和Newell [19]在七十年代初发明的环境映射方法在今天仍然很受欢迎，因为它易于实现。

Blinn-Newell方法巧妙地利用纹理，基本思想并不难理解。 环境（假定静态）的图像被捕获在单纹理或多个纹理中。 随后，用于绘制环境映射对象上的点的特定纹理和纹理坐标由观看者相对于对象的位置确定。

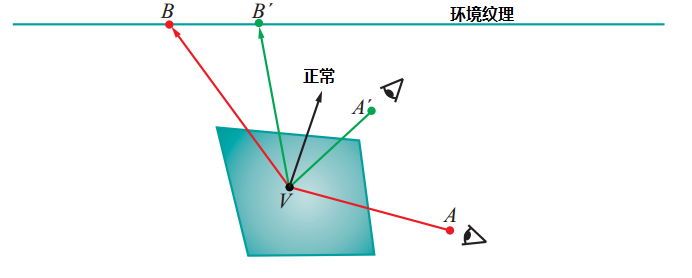


图13.14：Blinn-Newell环境映射原理：环境映射表面上的顶点V的纹理坐标可以从纹理图像上由源自眼睛的反射光线拍摄的点获得。

图13.14说明了原理。 环境映射四边形的顶点V处的纹理坐标由环境点确定——更确切地说，是观察者通过反射离开对象而看到的环境纹理的对应点。 例如，当观众处于A时，V以B的颜色值（图中的红色）涂漆; 当她移动到A’点时那些B’点被使用（绿色）。 Blinn-Newell方法的关键在于，随着视点的变化，基于反射定律动态计算纹理坐标。

OpenGL提供了两种环境映射方法的支持：球形映射和多维数据集映射。两者都基于Blinn-Newell方法，区别在于纹理上捕获环境和计算纹理坐标的方式。OpenGL为所有方法提供自动纹理坐标生成。 我们将详细讨论球形映射。

我们将把我们的演示文稿分为实现部分和理论部分，前者直截了当，是从业者最需要掌握的内容，而后者则更偏向理论且要求较高。

**13.6.1球面映射**

**让它工作**

使用OpenGL实现球面映射很简单，如下面的程序所示。



实验13.12。运行sphereMapping.cpp，它显示了一个飞船发射的场景，反射火箭锥最初固定在火箭前面的天空中。 按向上和向下箭头键移动锥体。 当锥体飞逝时，其发射图像表面上反射也变化。图13.15是一个飞船将要撞到地面的截图。

**结束**

这两个命令

glTexGeni（GL S，GL TEXTURE GEN MODE，GL SPHERE MAP）;

glTexGeni（GL T，GL TEXTURE GEN MODE，GL SPHERE MAP）;

在sphereMapping.cpp的初始化程序中，要求OpenGL使用其库中的函数来生成球面映射的s和t纹理坐标

绘图程序中的这对命令

glEnable（GL TEXTURE GEN S）;

glEnable（GL TEXTURE GEN T）;

和它的相反功能函数

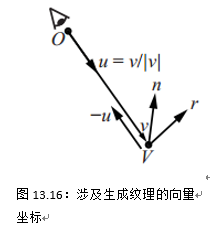
glDisable（GL TEXTURE GEN S）;

glDisable（GL TEXTURE GEN T）;

包围圆锥图，启用和禁用这些功能的使用。 所有这一切都是使用OpenGL实现球面地图！ 请注意，在球面映射被激活时刻绑定的纹理是发射图像，所以，这也是为什么发射图像被反映在锥体中。

现在，读者在看着锥体放大时可能会想知道反射到底有多真实。这是个很好的问题，它引导我们去弄清楚OpenGL如何计算球形映射的纹理坐标。

**如何工作**



这部分偏数学化。 如果您的兴趣只是那些实用的东西并且仅限于使用该技术，您可以放心地跳过它，并转到准备环境纹理的部分。

以下是在顶点V处生成球面映射纹理坐标的方法。 见图13.16。 来自眼睛的单位向量u（OpenGL中的原点O）朝向V是v / | v |，其中v是V的位置向量，当然，假设v = 0。那么来自V的单位眼睛方向是-u。V的单位正常值由用户提供。

OpenGL计算反射矢量r，在下列方程式的帮助下，通过以下方程式获得来自眼睛的假想射线的方向上的单位矢量（通过用练习11.4公式中的眼睛方向矢量-u替换光方向矢量u而获得））：

*r* = *u -* 2(*n · u*)*n*

假设OpenGL发现r =（rx，ry，rz）。下面计算的是数量

m =

最后，V处的纹理坐标计算为

和

但是，如果我们仔细地解析s和t的表达，那就不难理解游戏计划了。 使用上面的m表达式，记为

和

或者

和 (13.5)

其中变量为

和 (13.6)

一旦我们明白了映射

在几何学上，其余的东西将变得很好理解。

反射矢量r =（rx，ry，rz）是以原点为中心的单位球体S上的某点（例如P）的位置矢量。 见图13.17（a）。 现在，P相对于S的南极（0,0，-1）的位置矢量是r’=（rx，ry，rz + 1）。 而r’标准化后是矢量

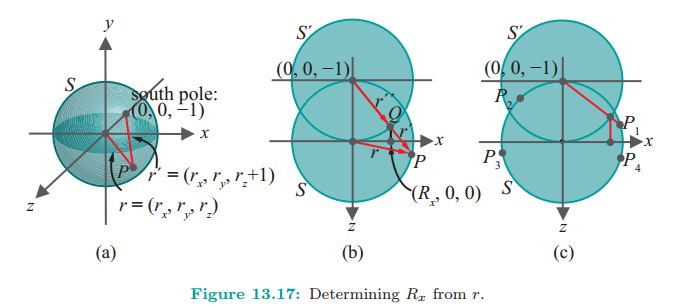


图13.17：从r确定Rx

其实r’’ 本身就是以南极为单位的位置向量，南极与P的交点Q以单位球S’’为中心。S’在图13.17（a）中没有画出，但图13.17（b）显示沿着xz平面的剖面中的S和S’（对于这个特定图，我们假定P位于本节）。Rx是Q的x值（13.7），Q在x轴上的投影如图13.17（b）

是（Rx，0，0）（Q的y值Ry是0，当然是在xz平面上）。 这是一个练习，以加强您对前述结构的理解，以从r中找到Rx。

**练习13.16**

对于图13.17（c）中的每个点Pi，1≤i≤4，使用标尺和铅笔在x轴上绘制相应的点（Rx，0，0）。 部分答案：红线表示P1的结构。

读者现在可以同意，至少当P在S的xz部分变化时，（Rx，0,0）在（-1,0,0）和（1,0,0）之间变化，并且相应地，Rx 在-1和1之间。此外，更靠近P到达南极，更靠近的是Rx到-1或1，这取决于极P的哪一侧。 然而，P不应该在南极，否则，确定Rx的建设是否崩溃。 因此，Rx本身既不达到-1也不达到1。实际上，考虑现在所有的球体S，而不仅仅是它的xz部分，不难看出，Rx在开放时间间隔（-1,1）上变化 P在S上减去其南极。

从P到Ry的映射是相似的。 因此，当P移动超过S减去其南极时，（Rx，Ry）在正方形[-1,1]×[-1,1]的内部移动。 为了更好的理解，我们来分析确定（Rx，Ry）对P的依赖性。

在-1 <Z≤1中选择Z。平面z = Z在纬度圆中与S相交

或者

现在，从（13.6）我们就有了

因此，如果P位于纬向圆，使得= 1-和= Z，则前面的等式表示（Rx，Ry）在圆上

(13.8)

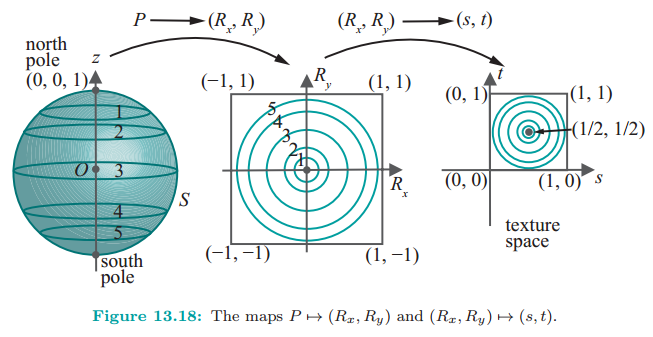
现在，我们可以看到（Rx，Ry）如何随着r =（,,）而变化。 事实上我们会画一张照片。 请参见图13.18左侧的两个图。

图13.18：映射P →（Rx，Ry）和（Rx，Ry）→（s，t）

请记住，r是P的位置向量，后者随S变化。S上的每个纬度圆（现在直立在左边，北极在顶部，以便更好地看到这些圆）映射到以起点和平方[-1,1]×[-1,1]为中心的圆 RxRy空间（在中间绘制）。特别是北极映射到原点，北极向北的纬度向下映射到[-1,1]×[-1,1]内越来越大的圆。已经绘制了五对相应的圆圈，并在两个图中类似地标出。当纬度线接近南极时，映射的圆越接近包含的方块。

**练习13.17**

赤道地图圆的半径是多少？纬度为45°N和60°S的图像的半径是多少？

提示：方程（13.8）给出了RxRy空间中的圆的半径，它是z = Z处纬线的图像。例如，纬度45°N具有z值sin45◦= 1 /，因此将Z = 1 /插入（13.8），以在RxRy空间中找到其映射圆的半径。

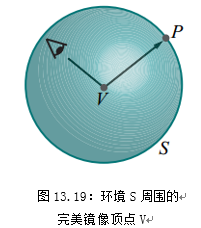
**练习13.18**

纵向大圆在S地图上怎么样？

提示：在RxRy空间中直线穿过原点

从RxRy空间到st空间（纹理空间）的最终转换很简单。见图13.18的最右边两个图。 （Rx，Ry）通过等式（13.5）映射到（1 2Rx + 1 2，1，2Ry + 1 2），其将RxRy空间中的平方[-1,1]×[-1,1]线性变换为 纹理空间中的单位平方[0，1]×[0，1]。 RxRy空间中五个圆的纹理空间中的图像也显示在最右边的图中。

**底线**



时间为简单的英语。 如果顶点V在一个完美的镜子上，周围的环境沿着以V为中心的单位球体S排列，那么眼睛将看到P与V的视线的反射相交的点P（见图13.19）。然而，OpenGL对环境的了解仅仅来自于纹理空间中占用单位正方形的用户提供的纹理。 那么它的作用是这样的：如果眼睛想要在球面环境中看到点P，OpenGL会代替P所映射的纹理空间中的点（s，t）显示它，如上所述P→（Rx， Ry）→（s，t）。

上面的计算准确地说明了物理方面发生了什么。 如果眼睛要看到环境的北极，那么它就会显示出纹理的中心。 随着眼睛的行进越来越远离北极点，它显示出距离纹理中心越远越远。 准确地说，环境中的纬度圈被替换为以其中间为中心的纹理中的圆圈以进行观察。

**准备环境纹理**

给定这种通过纹理呈现环境的球体映射方案，准备纹理的方法是什么？实际上，为了创建纹理图像，应该如何拍摄环境？比较图13.18的左图和右图表明了一个答案。 相机应位于原点O，朝向z轴朝向北极，并具有非常广角的镜头; 事实上，如果视野近360°，这将是有帮助的！当然，这是不可能的，但是使用位于要进行环境映射的对象附近的相机拍摄的相当广角的照片，并且重点放在世界空间的z轴上，应该会不错。

**备注13.4**

由于纹理使用仅取决于顶点反射矢量的值，而不是顶点的位置，平行方向的反射在所有顶点上表现相同。 实际上，这意味着与环境相比，环境映射的对象应该比实际的更小。

**备注13.5**

一些从业者主张在球形映射之前对纹理应用滤波器。 例如，NeHe [102]建议使用Spherizing过滤器（例如可以在Adobe的Photoshop软件中使用）。

**练习13.19（编程）**Sphere将鱼雷映射在shipMovie.cpp中。

只要您记住，CG真实性在于旁观者的眼中，就可以很好地尝试球形绘图。如果它看起来真实，那么它就是真实的。但是这仍然没有回答开始我们的数学调查的第一个问题：如何真实地反映sphereMapping.cpp的锥体？ 也许读者在这一点上已经有了自己的想法！

**13.6.2立方体映射**

立方体映射比球形映射更简单，通常也更真实。 实现它的唯一困难是必须在多个纹理

上捕获环境。 事实上，环境被想象成一个立方体，用户要求提供六个面中的每一个的图像。 图13.20给出了这个想法。

在立方体映射的情况下，给定反射矢量r，首先必须计算六个面中的哪一个被反射的光线，然后是实际点，从该位置的纹素提取颜色值。 对于球体映射，OpenGL中支持多维数据集映射。一旦用户加载六个环境纹理，OpenGL就会自动生成立方体映射的纹理坐标。我们将暂时搁置立方体映射，不再进一步讨论，将感兴趣的读者可以参考红色版的书。

图13.20：立方体映射