

硕 士 研 究 生 读 书 报 告



题目 基于OpenGL的纹理映射研究

作者姓名 肖梦婷

作者学号 21651161

指导教师 李启雷

学科专业 软件工程

所在学院 软件学院

提交日期 二○一七 年 四 月

Research of the Texture Mapping Based on OpenGL

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: The development of 5G mobile communication

Advisor: Li Qilei

By

Xiao Mengting

Zhejiang University, P.R. China

2017

摘要

OpenGL是指定义了一个跨编程语言、跨平台的编程接口规格的专业的图形程序接口。它用于三维图像，是一个功能强大，调用方便的底层图形库。

纹理映射，是将纹理空间中的纹理像素映射到屏幕空间中的像素的过程。简单来说，就是把一幅图像贴到三维物体的表面上来增强真实感，可以和光照计算、图像混合等技术结合起来形成许多非常漂亮的效果。在三维图形中，纹理映射的方法运用得最广，尤其描述具有真实感的物体。

然而纹理加载的过程可能会影响程序运行速度，当纹理图像非常大时，这种情况尤为明显。如何妥善的管理纹理，减少不必要的开销，是系统优化时必须考虑的一个问题。OpenGL硬件能够在内存中保存多个纹理，而不是每次使用的时候再加载它们，从而减少了运算量，提高了速度。

**关键词**：OpenGL，纹理映射，三维，真实感

Abstract

OpenGL refers to the definition of a cross programming language, cross platform programming interface specification professional graphical program interface. It is used for 3D image, is a powerful, easy to call the underlying graphics library.

Texture mapping is the process of mapping the texture pixels in the texture space to the pixels in the screen space. In simple terms, is to paste an image on the surface of three-dimensional objects to enhance the sense of reality, and can be combined with the calculation of light, image mixing technology to form a number of very beautiful results. In 3D graphics, texture mapping is the most widely used method.

However, the process of texture loading can affect the running speed of the program, especially when the texture image is very large. How to properly manage the texture and reduce unnecessary overhead is a problem that must be considered when the system is optimized. OpenGL hardware can save multiple textures in memory instead of loading them each time they are used, thus reducing the amount of computation and increasing the speed.

**Keywords：**OpenGL, texture mapping, 3D, sense of reality

# 1引言

在利用计算机对现实物体进行真实感绘制时，如果加入了光照，物体将会具有立体感，如果又给出了物体表面的材质、颜色等，物体将会变得丰富多彩并且还会具有一定的质感，从而使绘制出的图形具有了一定的真实感。但在有些情况下，仅仅使用材质属性等是不能获得更为真实的图形的，这时就需要使用纹理映射技术来绘制物体表面的每—个三维细节。

现实世界中的物体表面往往有各种纹理,因此要绘制逼真的三维物体,除了对物体进行光照处理外,还需要加上纹理贴图,使物体看起来更真实。在20世纪70年代末, 人们就已经开始发出了基于纹理映射的复杂算法。如环境映射、深度图和光照图。随着完全可编程硬件的出现,纹理映射及其应用越来越多。景物表面纹理的模拟在真实感图形显示中起着重要作用,纹理映射是一种提高真实感显示景物的技术。例如,对于表面凹凸不平的物体或具有纹理场景, 只使用材质属性是无法模拟景物表面的纹理特性的,这时就要使用纹理映射技术,即把具有表面纹理的图像粘贴到物体表面,从而获得具有真实感效果的物体模型。人们也正是用这些纹理特性来获取景物信息,感受景物形体形状以外的更多信息,如质感、色感等。

OpenGL 图形库除了提供基本的点、线、多边形的绘制函数外,还提供了复杂的三维物体(球、锥、多面体、茶壶等) 以及复杂曲线和曲面绘制函数,能够利用OpenGL 函数库实现各种三维物体的绘制,为实现三维物体的真实模拟必须引入纹理贴图技术,以便增强三维场景绘制的真实感,并能提高三维场景的渲染速度。

# 2 概述

## 2.1OpenGL简述

OpenGL 是跨平台的高性能3D 计算机图形AP I (Application Programming Interface )。它是由SGI公司推出的独立于操作系统和硬件环境的开放式三维图形库，作为一种用于实时3D 图形的工业标准，AP I已经得到了广泛的认可和接受。

OpenGL 提供了155个图形函数, 开发人员可以利用这些函数来构造景物模型，开发三维图形实时交互软件。OpenGL 具有强大的图形功能和良好的跨平台移植能力，目前已被广泛应用于可视化技术、实体造型、CAD /CAM、模拟仿真等诸多领域。纹理映射技术是OpenGL 的一个重要的技术，在进行模型转换和投影转换时都能执行操作,可用到所有的图元、点、线、多边形、位图和图像上。

OpenGL 是一个工业标准的三维计算机图形软件接口。它是一套独立于操作系统和硬件环境的三维图形库，有着强大的图形功能和良好的跨平台移植能力, 具有可靠度高、扩展性强、伸缩性好、灵活性强、容易使用等特点。OpenGL 能够生成真彩的三维场景,从绘制任何简单的3D物体到交互的动态场景, OpenGL

都能帮助用户高效地完成这些工作。具体地说，OpenGL的功能包括: 几何建模、坐标变换、颜色模式设置、光照和材质设置、图像效果、管理位图和图像、纹理映射、实时动画、交互技术等。

OpenGL 在Windows 平台上主要有以下几个图形库：

基本库opengl32. lib，头文件为gl. h，函数共有100 多个，都是最基本，最重要的函数，函数前缀为gl(如画点函数glVertex) ；

实用库glu32. Lib，头文件为glu. H，其中函数功能相对高级，如绘制复杂的曲线曲面，高级坐标变换，多边形分割(polygonal tessellation)。一共有40多个。前缀为glu(如NURBS曲面函数gluNurbsSurface)；

辅助库glaux.lib，头文件为glaux.h，是一些鼠标、键盘控制、窗口管理等简单的交互函数和常用三维物体的绘制函数。一共有30多个。前缀为aux

(如窗口初始化函数auxInitWindow)。由于简单，大型的OpenGL 程序很少使用到这一套函数。近年来，OpenGL 的编程者比较多地使用到一个叫实用工具库glut32. lib，头文件glut. h。辅助库可以看作是实用工具库的一个子集。工具库还提供了许多辅助库没有的功能，如菜单操作等。前缀为glut(如菜单创建函数glutCreateMenu)。这个库不是MFC自带,由SGI 在OpenGL 的官方网站上提供。

此外,还有六个WGL函数非常重要,专门用于OpenGL 和Windows 95窗口系统的联接,其前缀为wgl ,主要用于创建和选择图形操作描述表(rendering contexts)以及在窗口内任一位置显示字符位图。这些功能是Windows 95对OpenGL 的唯一补充。一个OpenGL 的Windows95 应用程序的基本操作步骤如下:创建OpenGL 的着色环境包括设定象素格式,创建着色环境,把着色环境指定为当前的着色环境;进行OpenGL相关的初始化,如设立灯光,质材,纹理,变换等等;绘制图形；将着色环境设为非当前使用,释放着色环境。

## 2.2纹理简介

纹理的定义有两种：连续法和离散法。连续法把纹理定义为一个二元函数,函数的定义域就是纹理空间。而离散法则是把纹理定义在一个二维数组中,该数组表示纹理空间中行间隔和列间隔固定的一组网格点上的纹理值。网格点之间的其它点的纹理值可以通过对相邻网格点上纹理值进行插值来获得。通过纹理空间与物体空间之间的坐标变换，可以把纹理映射到物体表面。一般来说，离散法是较为常用的纹理定义方式。

根据纹理的表现形式可以分为颜色纹理、几何纹理和过程纹理。

(1)颜色纹理

颜色纹理是通过颜色色彩或明暗度的变化体现出来的表面细节，它在光滑表面上描绘附加定义的花纹或图案。在物体表面绘制图案，就是改变物体表面相关部分的反射和透射系数。颜色纹理的实质是将一个二维纹理模式映射到物体表面上，二维纹理模式定义在一个平面区域上，平面区域的每一个点定义一个灰度或颜色值,该平面区域称为纹理空间，该方法称为纹理映射。

(2)几何纹理

几何纹理是指基于景物表面微观几何形状的表面纹理，它是根据粗糙表面的光反射原理，通过一个扰动函数扰动物体表面参数，使表面呈现出凹凸不平的形状。

Blinn在1978年提出, 用扰动物体表面法线方向的方法模拟表面凹凸纹理的真实感显示效果，称为凹凸映射。其基本原理是：对原物体表面的法线方向附加一个扰动函数，由于表面光亮度是景物表面法向量的函数，这法向量的扰动导致表面光亮度的突变，使得原来法线方向的光滑、缓慢的变化方式变得剧烈、短促, 通过光照形成表面凹凸粗糙的显示效果。

(3)过程纹理

用一些简单的可解析的数学模型来描述复杂的自然纹理细节,即用过程方式将纹理空间的值映射到景物的表面,生成的三维纹理称为过程纹理。Perlin 在1985年,提出一种近似描述湍流现象的经验模型，用一系列的三维噪声函数的叠加来构成湍流函数，表现了各种规则或不规则的动态变化的自然景象，如水波、云、火、烟雾等。

## 2.3纹理映射技术介绍

纹理映射技术是计算机图形学中广泛应用的一项重要技术。传统的几何造型只能表示景物的形状，无法描述景物表面的微观细节，而利用纹理图像来描述景物表面各点处的反射属性，可以达到模拟景物表面丰富的纹理细节的目的，提高计算机生成图形的真实性。另一方面，采用纹理映射的方法可以大大地简化建模的过程。比如，同是一栋大楼表面，若完全用建模的方法来构造，则需要画出大楼的每一扇门、每一扇窗户；若采用纹理映射，只需建立简单的长方体模型，用拍下的大楼外观图片贴至模型表面即可。相比之下，工作量要小得多。

纹理映射是通过将像素与几何对象结合，为图像营造出一种非常复杂的视觉效果，避免了建立大量几何模型所需要的开销。通常我们在实现屏幕上显示对象的方法，一种是建立该对象的几何模型（通常使用三维多边形建模），然后将其送入几何流水线；另一种是用像素块来表示对象。但两种方式都有其自身的局限性；几何流水线系统在使用大量几何单元构筑的基元时，系统处理速度会大幅度下降，难以满足需求；像素块可以较好的表现局部细节，却不具备三维属性．纹理映射的方法就是将两种方法最突出的特性结合起来，同时使用几何流水线和像素流水线。

纹理贴图类型主要有材质贴图、Mip贴图、凹凸贴图、动态材质贴图等。材质贴图是将一张已有的图像直接贴到物体的表面。Mip贴图依据不同的精度要求，使用不同版本的材质图像进行贴图。凹凸贴图是一种在三维场景中模拟粗糙表面的贴图技术。它将深度的变化保存在一张贴图中，然后将对三维物体进行混合贴图处理，最后生成具有凹凸感的贴图效果。动态材质贴图技术可产生较好的材质贴图效果。

# 3 基于OpenGL纹理映射实现

要使用纹理映射,可在应用程序中执行如下步骤:

## 3.1使用glGenTextures()获得未用的纹理对象标识符

要获得未用的纹理对象标识符, 可使用glGenTextures()。不再需要纹理对象标识符时,可使用glDeleteTextures()将其归还。

void glGenTextures(GL sizei n , Gluint3 textures);

void glDeleteTextures ( GL sizei n , const Gluint 3 textures);

glGenTextures()返回n个未用的纹理对象标识符，将其标记为已用，并将他们存储在textures中。glDeleteTextures删除为textures中的n 个标识符相关联的纹理对象,并将这些标识符归还给未用纹理标识符池。

## 3.2设置纹理对象的状态参数

使用命令glTexParameteri()来设置纹理对象的状态。

viod glTexParameteri[if]( Glenum targe , Glenum pname ,TYPE param ) 。

## 3.3使用glTexImage2D()或glutBuild2DMipmaps()指定纹理图像

首次绑定某个纹理对象并设置其状态时,还应调用glTexImage2D()来指定纹理图像。如果应用程序将GL-TEXTURE-MIN-FILTER设置为mipmap 模式,必须指定mipmap 金字塔中所有的纹理图像。可以将纹理参数GL-GENERATE-MIPMAP设置为GL-TRUE, 然后再使用glTexImage2D()来指定纹理,如:

glTexParameteri ( GL - TEXTURE - 2D , GL - GENER2ATE - M IPMAP , GL - TRUE);

glTexImage2D ( , , , ) ：

如果GL-GENERATE-MIPMAP被设置为GL -TRUE, glTexImage2D ( ) 对纹理图像进行滤波, 生成一系列明细度更低的图像, 以组建整个mipmap 金字塔。如果以后的修改影响了基本纹理图像, OpenGL 将重新计算mipmap 图像。

在不支持GL-GENERATE-MIPMAP的版本中使用GLU 函数glutBuild2DMipmaps ( ) 来创建所有的mipmap 等级。

void glTexImage2D ( Glenum target, Glint lever, Glint in2ternalformat, Glsizei width , Glsizei height, Glint border, Glenumfomat, Glenum type , const Glvoid 3 data ) 。

## 3.4渲染使用纹理对象的几何体之前,使用glBindTexture()绑定纹理对象

glGenTextures()返回纹理对象标识符后, 要使用glBindTexture()命令来激活纹理对象, 创建与标识符相关联的纹理对象。

Void glBindTexture ( Glenum target, Gluint texture )。

## 3.5渲染几何体之前启用纹理映射

## 3.6将几何体发送给OpenGL , 并给每个顶点指定合适的纹理坐标

当前纹理已经存在之后,就可以给顶点指定纹理坐标,以说明这一顶点的纹理像素在图上的位置。OpenGL会对根据顶点坐标对平面内部进行分割,以确定每一点对应纹理图上的那个像素。

指定当前纹理坐标的命令是glTexCoord ()，此后定义的所有顶点的纹理坐标都会是( s, t, r, q ) 。( s, t ) 是顶点纹理相对图片左下角原点的位置, 坐标值应在0. 0～1. 0 之间。也可以给顶点指定比1 大比0 小的纹理坐标,对于这种超出纹理图范围的情况,可以用

glTexParameterf ( GL-TEXTURE-2D , GL-TEXTURE-WRAP-S, GL-REPEAT) ;

glTexParameterf ( GL-TEXTURE-2D , GL-TEXTURE-WRAP-T, GL-REPEAT) ;

glTexParameterf ( GL-TEXTURE-2D , GL-TEXTURE-WRAP-S, GL-CLAMP ) ;

glTexParameterf ( GL-TEXTURE-2D , GL-TEXTURE-WRAP-T, GL-CLAMP ) ;

选择:

GL-REPEAT: 当纹理比表面小时重复使用纹理以填满每个点。

GL-CLAMP: 比1 大的当作1 ,比0 小的当作0。

对于复杂的物体表面来说逐一指定其纹理坐标是相当烦琐的事,如使用纹理映射构造模型的轮廓图, 或者模拟光亮物体表面反射周围场景中的景物效果时, 可用glTexGen ( ) 函数自动生成纹理坐标。这个函数自动生成某一方向上的纹理坐标。

viod glTexGen { ifd } { v} ( Glenum coord , Glenum pname ,TYPE param ) 。

# 4 纹理映射的应用

## 4.1人脸仿真

为实现三维特定人脸逼真纹理映射, 首先需对欲映射的人脸图像按照人脸的特定部位进行特征点提取。人脸的特征部位分别定为: 左右眉毛、左右眼睛、鼻子、上下嘴唇、正面的左中右边界线。然后根据三维人脸相应部位的网格点计算欲映射图像的纹理坐标, 最后调用OpenGL纹理映射函数进行三维人脸逼真纹理映射。三维人脸逼真纹理映射过程如图4.1所示。该应用将人脸区域分成轮廓较平坦区域和轮廓凹凸明显区域两类, 分别设计了两种不同的算法进行纹理坐标的计算。

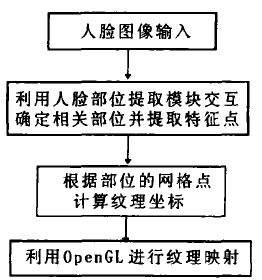


图4.1 纹理映射过程

（1）人脸特定部位特征点提取

首先对原始照片交互选取脸部图像, 如图4.2所示。接着, 交互框出脸部特征部位的范围, 如额头、眉毛、眼睛、脸颊、鼻子、嘴唇、下巴等, 如图4.3所示。在框出特征部位的基础上, 进一步逐一提取特征点, 如图4.4所示。



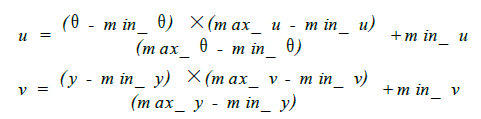
图4.2 脸部图像 4.3 特征部位选取 4.4 特征点选取

（2）三维人脸网格点纹理坐标计算

人脸按功能可分成两类不同的区域, 一类是比较平坦的部位(如脸颊, 额头等) ; 另一类是凹凸较明显的部位(如鼻子、嘴唇、眼睛等) 。平坦部位的纹理映射较简单,不容易发生纹理映射错位的现象。对于凹凸明显的部位, 由于在照片上标注的特征点与三维人脸模型上的网格点的数目不一致, 一般是人脸模型上标注的特征点的个数多于纹理照片上特征点的个数,因此需要通过插值的方法得到那些没有纹理对应的三维人脸模型上网格点的坐标。

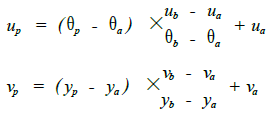
首先将人脸模型的每一个网格点的三维坐标( x, y, z) 通过坐标变换使之成为柱面坐标(θ, y) 。对于较平坦部位的网格点的纹理坐标的计算,步骤为:

1. 从所选取的特征部位的所有已知纹理坐标中, 求得最大最小纹理坐标值:m ax\_ u, m ax\_ v, m in\_ u, m in\_ v；
2. 对应于三维人脸模型中相应要映射的部位,求得三维人脸模型上该对应部位的坐标值的范围,设其最大最小坐标值分别是:m ax\_θ, m ax\_ y, m in\_θ, m in\_ y；
3. 采用式下列公式计算与该特征部位三维人脸特征点坐标(θ, y) 对应的纹理坐标( u,v) :



综上, 平坦部位中的任意三维坐标所对应的纹理坐标即可确定。

至于轮廓较明显的部位,首先,应根据选取的照片纹理特征点准确计算出三维部位轮廓特征点的纹理坐标。已知纹理坐标特征点构成一个凸多边形。如图4.5所示,线段ab和线段bc是已知纹理坐标的连线, a点的纹理坐标为( ua , va )，b点的纹理坐标为( ub , vb )。首先结合该部位θ方向的最大最小值m ax\_θ，m in\_θ和纹理坐标u方向的最大最小u值m ax\_ u, m in\_ u,求出a, b两点纹理坐标所对应的三维坐标中的坐标点(θa , ya )，(θb , yb )，然后依下列公式计算出轮廓网格点p (θp , yp ) 的纹理坐标( up ,vp )：



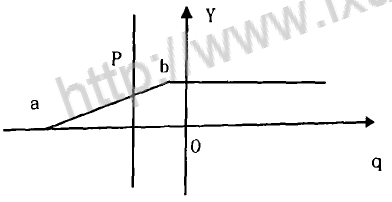
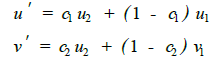


图4.5 轮廓特征点纹理坐标

考虑到照片上交互得到的标注的特征点个数少于三维人脸模型上依据M PEG - 4标准标定的特征点的个数,因此三维人脸模型上无纹理对应的特征点的纹理坐标需要通过对那些已知纹理坐标值的特征点纹理坐标进行线性插值来得到。这个插值过程如图4.6所示, 其中●表示已存在纹理对应的轮廓点,如点p1 (θ1 , y1 ) 和点p2 (θ2 , y2 ) ; ○表示尚无纹理对应的轮廓点, 如点p3 (θ3, y3 ) 。可利用如下所示插值公式得到p3 (θ3 , y3 ) 的纹理坐标( u′, v′) 。这样就可依次得到某选定区域的所有轮廓点的纹理坐标。



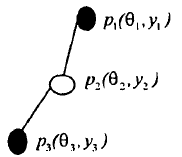


图4.6 轮廓点纹理坐标插值计算示意图

对于凹凸明显区域内的网格点, 其纹理坐标同样可以采用线性插值方法求得。如图4.7所示, 其中:●表示该部位的轮廓点纹理坐标已经生成, ○表示部位内的未生成纹理坐标的网格点。可在相应的逻辑网格中找出部位的边界点, 这些边界点均已具有纹理坐标, 因此同样可采用插值公式计算出这些部位内网格点的纹理坐标。

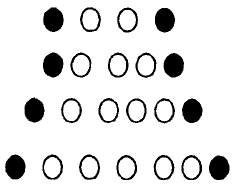


图4.7 轮廓内的网络点的插值示意

综上所述, 通过对平坦区域和凹凸明显区域采用不同的纹理坐标计算方法, 得到三维人脸模型上每一网格点的纹理坐标。

（3）基于Op enGL的纹理映射实现

1）纹理定义

采用二维纹理, 在OpenGL中的实现函数是glTexImage2D ( ) [ 6 ] ,

2）纹理控制

OpenGL通过glTexParameter ( ) 函数对纹理进行控制, 即得证纹理图像的纹理与屏幕上的像素对应, 通过纹理贴图实现纹理缩限和纹理重复等。

3）纹理贴图方式

在进行纹理贴图时, 纹理图像可以直接贴到物体上从而覆盖物体已有的颜色, 也可以和物体原有颜色进行融合。融合是一种重要的图像处理技术,主要用于透明效果处理。OpenGL用于纹理贴图方式的函数为glTexEnv ( ) 。

4）定义纹理坐标

许多信息是通过顶点来传递的。顶点的信息有几何坐标、顶点法向量、顶点颜色等等, 但与顶点相关的另一类重要的图像控制信息是顶点的纹理坐标。纹理坐标控制纹理图像中的纹素如何映射到物体上。

# 5 小结

纹理映射是OpenGL 开发过程中非常重要的环节之一。纹理映射是真实感图像制作的一个重要部分，运用它可以方便的制作出极具真实感的图形而不必花过多时间来考虑物体的表面细节。

本文主要介绍了OpenGL 纹理映射的基本概念、基础技术以及具体应用，包含纹理的定义、纹理映射技术的原理和实现、纹理映射在人脸仿真中的应用对系统掌握OpenGL 纹理映射技术具有一定的意义。

参考文献

[1]Li Cui, Hyungyu Kim, Euee S. Jang:A Hybrid Texture Coding Method for Fast Texture Mapping. JCSE 10(2) (2016).

[2]Yanning Wang, Heming Sha, Weiyao Lin, Hua Yang, Chia-Wen Lin:Improved 3D-model texture mapping with region-of-interest weighting and iterative boundary-texture updating. ICME Workshops 2016: 1-6.

[3]Daniele Panozzo, Olga Diamanti, Sylvain Paris, Marco Tarini, Evgeni Sorkine, Olga Sorkine-Hornung:Texture Mapping Real-World Objects with Hydrographics. Comput. Graph. Forum 34(5): 65-75 (2015).

[4]Minerva Singh, Damian Evans, Daniel A. Friess, Boun Suy Tan, Chan Samean Nin:Mapping Above-Ground Biomass in a Tropical Forest in Cambodia Using Canopy Textures Derived from Google Earth. Remote Sensing 7(5): 5057-5076 (2015).

[5]Charalampos Koniaris, Kenny Mitchell, Darren Cosker:Real-time variable rigidity texture mapping. CVMP 2015: 5:1-5:10.

[6]Peter Cheng, Michael Anderson, Stewart He, Avideh Zakhor:Texture mapping 3D models of indoor environments with noisy camera poses. Computational Imaging 2014: 90200V.

[7]Songgang Xu, John Keyser:Texture mapping for 3D painting using geodesic distance. I3D 2014: 163.

[8]Yanlin Weng, Dongping Li, Yiying Tong:Texture mapping subdivision surfaces with hard constraints. The Visual Computer 29(11): 1231-1241 (2013).

[9]Mohammed Elfarargy, Amr Rizq, Marwa Rashwan:3D Surface Reconstruction Using Polynomial Texture Mapping. ISVC (1) 2013: 353-362.