第三章 计算机图形学理论基础与OpenGL图形库

3.1 引言

前文介绍了等几何拓扑优化的理论知识，并基于理论利用matlab生成得到了各个单元体所对应的设计变量（即密度值）信息。在开发可视化系统实现等几何拓扑优化结果的过程中，图形学理论是核心知识需求。为了准确地将数据显示为图形，并为后文实现GPU渲染加速提供理论基础，本章对图形学理论基础以及OpenGL图形库进行了介绍。

3.2 图形学基础

计算机图形学是研究计算机在硬件和软件的帮助下创建计算机图形的科学学科，是计算机科学的一个分支领域，主要关注数字合成与操作视觉的图形内容。在本文中，计算机图形学主要运用于等几何拓扑优化结果的图形显示，这体现了计算机图形学涉及到计算机领域中的数据结构与算法、数学领域中的线性代数等学科知识。本节主要针对图形学中的图形几何变换以及常见的坐标系统进行介绍。

3.2.1 坐标系统

OpenGL希望在每次顶点着色器运行后，我们可见的所有顶点都为标准化设备坐标(Normalized Device Coordinate, NDC)。也就是说，每个顶点的x，y，z坐标都应该在-1.0到1.0之间，超出这个坐标范围的顶点都将不可见。我们通常会自己设定一个坐标的范围，之后再在顶点着色器中将这些坐标变换为标准化设备坐标。然后将这些标准化设备坐标传入光栅器，将它们变换为屏幕上的二维坐标或像素。对我们来说比较重要的总共有5个不同的坐标系统：局部空间（Local Space）、世界空间（World Space）、观察空间（View Space）、裁剪空间（Clip Space）、屏幕空间（Screen Space）。

为了将坐标从一个坐标系变换到另一个坐标系，我们需要用到几个变换矩阵，最重要的几个分别是模型（Model）、观察（View）、投影（Projection）三个矩阵。我们的顶点坐标起始于局部空间，在这里它称为局部坐标，它在之后会变为世界坐标，观察坐标，裁剪坐标，并最后以屏幕坐标的形式结束。下面的这张图展示了整个流程以及各个变换过程做了什么：



图2 坐标系统转换流程

3.2.2 图形变换

3.2.2.1齐次坐标

三维数据可以通过三维向量与3×3矩阵的乘法操作，来完成缩放和旋转的线性变换。在大多数3D工作中，我们参照的依据是几何学中的三维空间（X, Y, Z）。但在某些情况下，参照投影几何更适用，除了X, Y, Z分量外，增加一个W分量，这个四维空间叫做“投影空间”，在四维空间中的坐标叫“齐次坐标”。

齐次坐标是计算机图形学的重要手段之一，它既能够用来明确区分向量和点，同时也更易用于进行仿射（线性）几何变换。本小节以三维变换矩阵举例，三维齐次坐标相当于四维空间中的坐标，由此三维图形几何变换通过利用齐次坐标可将其放置在四维空间中进行，以下为4×4方阵为三维变换矩阵：



3.2.2.2 平移、旋转、缩放

在三维坐标系中，平移、旋转、缩放变换如图

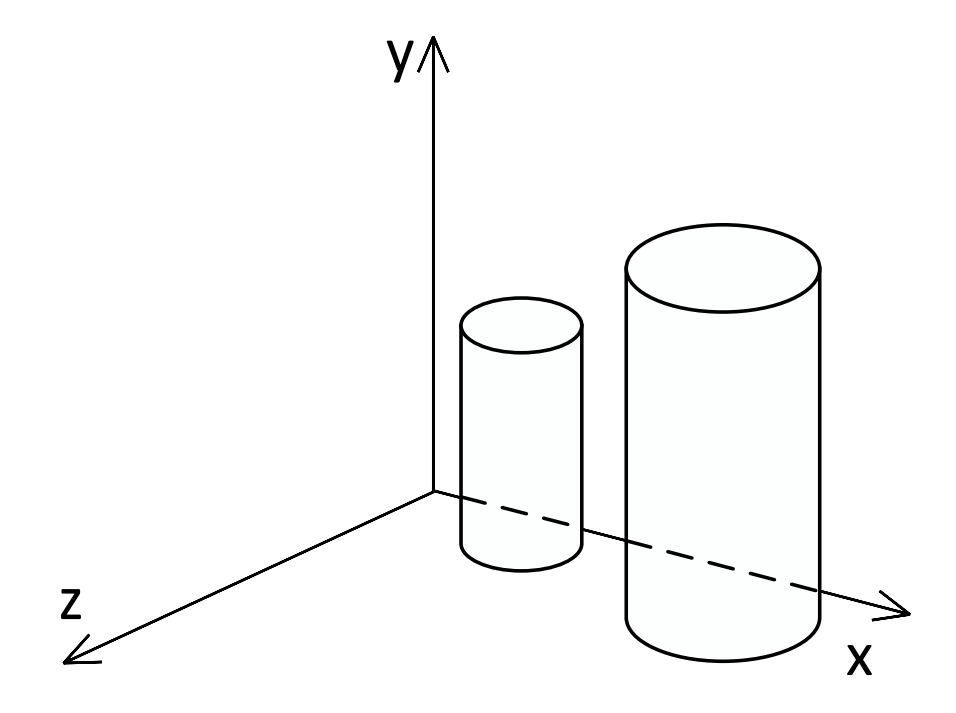
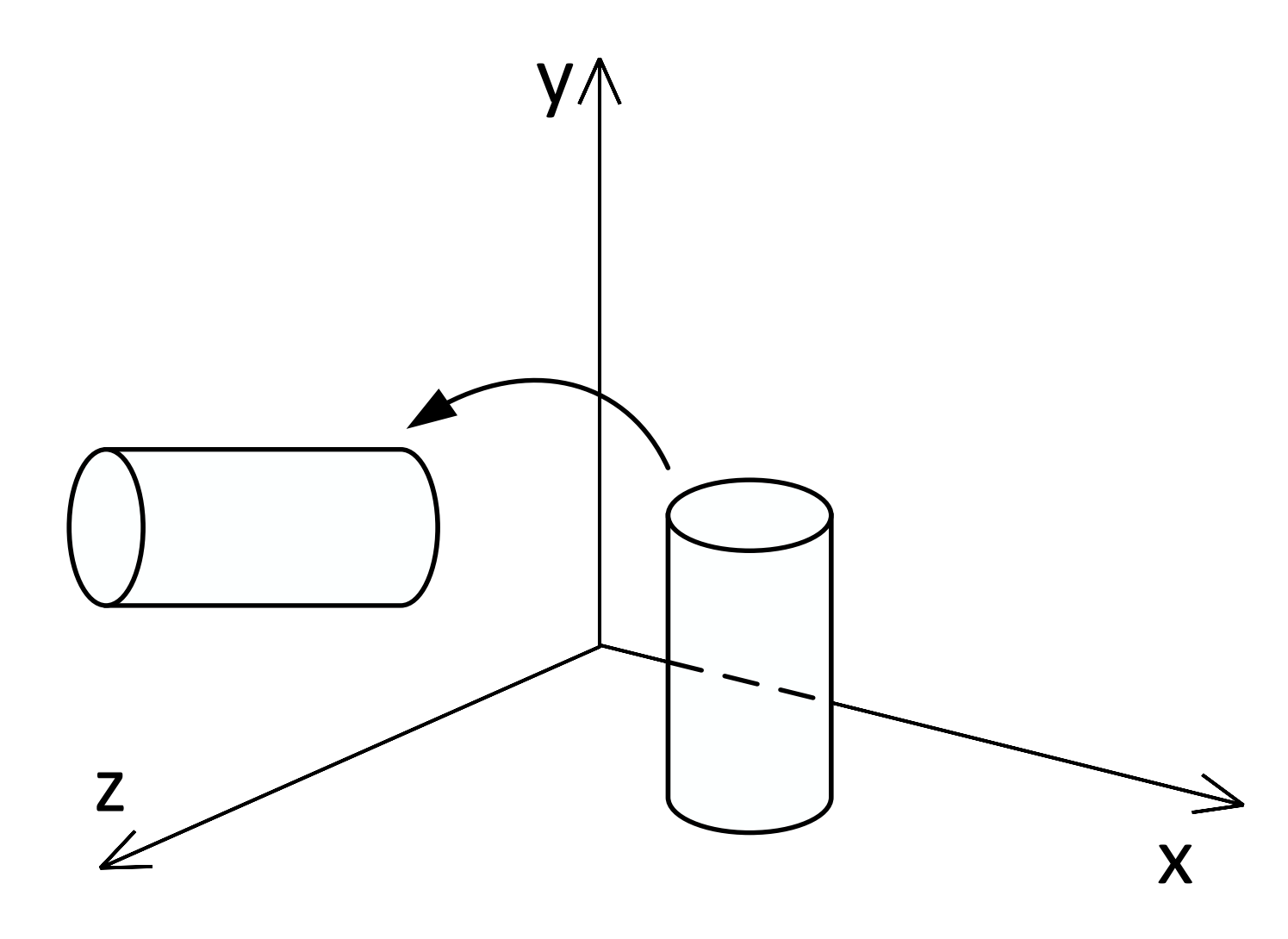
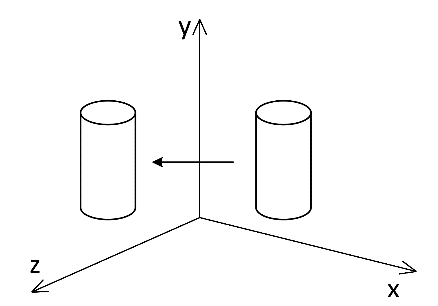


图3

1、平移变换

平移变换的相应矩阵运算过程表示如下：



因此三维平移变换矩阵为：



2、旋转变换

图形对象绕空间中任意轴旋转可被视为绕各坐标轴旋转的组合，如图3为绕z轴旋转。

经过旋转后的图形坐标表示为，因此三维旋转变换矩阵为：



3、缩放变换

图形缩放后的坐标可由原始坐标表示为，因此三维缩放变换矩阵为：



3.2.2.3 观察矩阵

观察矩阵（View Matrix），就是将物体在世界空间的坐标变换到观察空间的矩阵。观察空间需要一个观察者作为参考对象，一般来说都是摄像机。摄像机在世界空间内某个坐标观察场景中的物体，观察空间就是摄像机看到的视图空间。假设在一个世界空间内，有一个摄像机（即观察者）摆放在某个位置，可以通过将世界空间的坐标乘以观察矩阵来得世界空间内的物体相对于摄像机的位置，简而言之，就是转换得到摄像机看到的世界。

将摄像机从世界空间中某个坐标移动到原点，再将摄像机坐标abc进一步旋转变为标准坐标系XYZ的过程，即为摄像机变换的过程，即观察矩阵由平移矩阵和旋转矩阵相乘组合而来，表达式如下：



等价地，也可以利用LookAt矩阵进行摄像机变换。LookAt()是OpenGL中提供的一个函数，其作用就是通过传入摄像机位置、目标位置和摄像机的向上向量即可得到观察矩阵。LookAt矩阵表达式如下：



其中*R、U、D*分别为右向量、上向量和方向向量，即相机坐标系*x、y、z*在世界坐标系中的表达；*P*为相机在世界坐标系中的位置。

3.2.2.4 投影矩阵

在一个顶点着色器运行的最后，图形接口期望所有的坐标都能落在一个特定的范围内，且任何在这个范围之外的点都应该被裁剪掉。这也就是裁剪空间名字的由来。为了将顶点坐标从观察变换到裁剪空间，我们需要定义一个投影矩阵，它指定了一个范围的坐标，接着会将在这个指定的范围内的坐标变换为标准化设备坐标的范围，在这一阶段之后，最终的坐标将会被映射到屏幕空间中，并被变换成片段。

将观察坐标变换为裁剪坐标的投影矩阵可以为两种不同的形式，即正交投影矩阵或一个透视投影矩阵。

1、正交投影

设相机距离近裁剪平面的距离为*near*，距离远裁剪平面的距离为*far*，横纵比为*Aspect*，视锥体高度的一半的大小为*size*。正交矩阵需要将



均映射到。其中*z*是反向映射，即**映射到1，映射到，这是因为观察空间为右手坐标系，而OpenGL的NDC为左手坐标系。OpenGL的正交投影表达式如下：



2、透视投影

设相机距离近裁剪平面的距离为*near*，距离远裁剪平面的距离为*far*，横纵比为*Aspect*，垂直方向的张开角度为*FOV*。投影变换首先将顶点映射至近裁剪平面，然后将近裁剪平面上的顶点的*x*和*y*映射到。OpenGL的透视投影表达式如下：



3.3 Win32与OpenGL程序框架

3.3.1 OpenGL程序框架

作者编写的OpenGL的基本框架由如下四个函数组成：

（1）InitScene()：在该函数中实现整个应用程序的初始化，包括窗体初始化、OpenGL初始化、主要数据初始化等工作，并在该阶段中进行首次NURBS细分曲面的计算，将计算着色器得到的数据与相应的OpenGL缓冲区绑定，在之后的渲染过程中进行数据在屏幕视口的显示。

（2）UninitScene()：在图形程序退出时进行调用，断开程序与OpenGL的连接，清理所有C++生成的指针等信息，防止内存泄漏等问题的出现。

（3）Update()：在渲染步骤前调用，主要进行摄像机的更新以及显示图形形状、状态、显示模式等属性的更新。

（4）Render()：渲染过程，是利用OpenGL进行图形显示所需的关键函数，再整个应用程序正常运行的时间内，都会根据需要不断进行调用，并在调用函数的过程中给出所有关于图形显示的命令。

