**虎书总结**

# 引言

简单介绍一些图形领域，图形学的主要应用，图形用户接口，图形管线，数值问题，代码的效率，图形程序的设计与编程，图形程序的调试。

注意点：

常常把调试输出用图片的形式显示

# 微积分

集合与映射，对数函数的运算，二次方程求解，三角函数，向量，坐标系，曲线曲面，线性插值，三角形

注意点：

1. 从一个向量或者两个向量重建笛卡尔坐标系
2. 三角形中重心坐标 p46

# 栅格化图像

栅格化设备：一些输入输出设备；RGB颜色；alpha分量；

注意点：

1. 图像位深度 p61
2. 显示器显示的值与图像的值：非线性、量化 p62

# 光线追迹

基本过程；投影；光线投射；光线相交；光线渲染；阴影；

注意点：

1. 两种渲染形式：逐物体渲染、逐像素渲染 p69
2. 光线追迹的基本过程：光线投射、光线相交、光线的渲染 p70
3. 光线（射线）的表示方法 p73
4. 两种投影方式决定了两种光线投射方式 p74
5. 光与物体相交性：p76

光线与球相交：（化为一元二次方程，判断根存在性）

光线与三角形相交：（联立求解三元一次方程，克拉默法则）

光线与多边形平面相交：（先求交点，再判断是否在内部）（或划分为多个三角形）

1. 求交点：
2. 判断是否在内部：先将3D多边形投影到坐标2D平面（根据法向分量判断），从相交点往一个坐标轴方向发出一条线，求之与2D多边形边的交点，以此来判断内部与否。
3. 光线的渲染：p81

单根光线的渲染：（环境光+漫反射+镜面反射）

多根光线，叠加即可

1. 光线追迹的面向对象编程：建立合适的类来管理 p85
2. 阴影：p86

从一个点往光源发出射线，判断途中是否被挡（相交性检测），一个细节：而不是从0开始，因为有可能由于精度问题，而误判与点所在平面相交

1. 镜面反射的多次追踪，从相交点朝反射方向发出光线再做追迹 p87
2. 光线追迹的其他用途：判断鼠标点击的某一物体是哪个物体？p88

# 线性代数：

行列式；向量；矩阵；特征值；奇异值分解

注意点：

1. 奇异值分解：（U，V为正交阵，S为对角阵）p103
2. 求解奇异值分解：（分别对和进行本征值分解：

# 变换矩阵

缩放；剪切；旋转；变换矩阵的组合与分解

注意点：

1. 2D缩放：p110
2. 2D剪切：（相当于倾斜坐标轴：y轴倾斜）p111
3. 2D旋转：p112
4. 2D反射：（相当于旋转180度）p114

由于一个矩阵就可以代表几个组合的变换，同时根据奇异值分解（正交阵看作旋转，对角阵看作缩放），所以任意变换均可看作：旋转-缩放-逆旋转。

1. 在栅格化图像中，把旋转可看作三个剪切（剪切相当于一时只对一维数据操作，速度更快）p122
2. 3D旋转：（绕z轴）p124
3. 3D中沿任意轴旋转：根据旋转轴建一个正交直角坐标系uvw空间，故先转到uvw空间，再旋转，再转回xyz空间 p125
4. 若一个向量及其法向量同时做变换，要使变换后仍然垂直，则两个变换矩阵互逆 p127

仿射变换：平移变换不能简单表示，需要用齐次矩阵。注意点与方向的齐次表达区别： p129

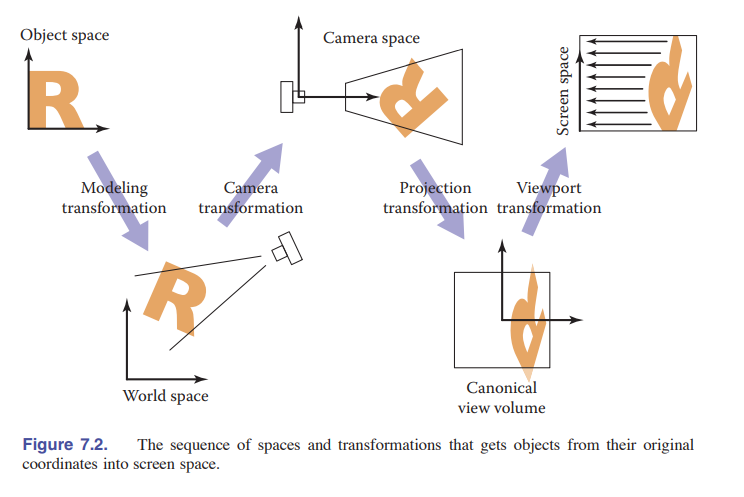
1. 求变换矩阵的逆矩阵：p132
2. 对角阵
3. 正交阵

一般矩阵，奇异值分解

# 视点变换

注意点：

1. 视点变换在逐物体渲染中，发挥重要作用 p139
2. 从物体坐标系到屏幕坐标系的流程图：p141

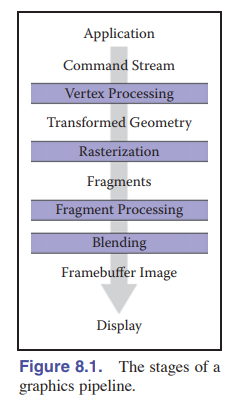


注意这个过程中：lookat矩阵，投影矩阵（两种），视锥体

# 图形管线

注意点：

1. 图形管线的概念：从物体到图像像素的过程 p159
2. 图形管线的大致流程：输入顶点信息——处理顶点（经过矩阵变换到屏幕坐标系）——光栅化——处理片元——混合片元——存到FrameBuffer——显示



1. 光栅化：p160

线段的光栅化：一维遍历

三角形的光栅化：依赖重心坐标，二维遍历（可用累加来优化）

多边形的光栅化：扫描线算法或者拆成三角形

消除洞与重复绘制：任意取屏幕外一点，同一侧的才画

1. 裁剪问题：

线段ab的裁剪：线段与平面的裁剪 p171

1. 逐顶点渲染与逐片元渲染：p175

逐顶点渲染：称为Gounraud shading，先计算顶点的颜色，中间的片元颜色插值得到

逐片元渲染：成为Phong shading，先插值法线等，再计算中间的片元颜色

1. 简单的抗锯齿：用box滤波器 p178
2. 剔除问题：注意与裁剪区别 p179

三种情况：

1. 视锥剔除：视锥体外；
2. 遮挡剔除：视锥体内但被遮挡；
3. 背向剔除：视锥体内但背向（被自己另外的三角形挡）

# 信号处理

信号采样与重建；卷积；卷积滤波器；图像中的信号处理；采样理论；卷积与傅里叶变换；

1. 重采样问题：p212

先重建一个连续函数，随后再采样，为避免锯齿，两个过程都要用合适的滤波器。滤波器需要在速度和质量上权衡，按质量排序为：box——tent——piecewise cubic，同时为了加速，常常把行列分开进行操作。

# 表面渲染

漫反射；环境光；Phong模型（区别于Phong着色）；艺术渲染、

# 纹理映射

1. 概念：映射图像到表面上；应用：制造阴影和反射，定义表面的微结构

两个难点：容易变形、重采样带来的锯齿 p243

1. 纹理映射的两个主要问题：建立纹理坐标方程、查找纹理值而不引入严重的锯齿 p246
2. 纹理坐标方程的条件：双射性、连续性、偏导大小在距离和方向上平滑 p247
3. 纹理坐标方程建立的两个方法： p248

从表面点出发在几何上计算方程

1. 垂直投影和透视投影：
2. 球坐标映射，根据来映射，忽略
3. 柱坐标映射
4. 立方体映射

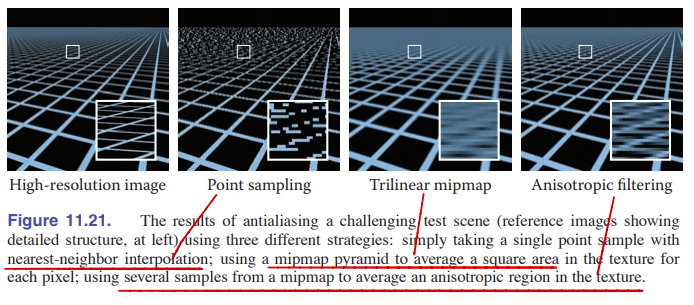
在顶点中存纹理坐标插值使用：

1. 纹理映射的质量取决于网格体在纹理上的排布。
2. 对于边缘有多种处理：设置背景色、clamp取边缘色、取余重复
3. 要得到正确的透视纹理映射，需要修正：
4. 为了解决接缝处的连续性：在接缝处，复制纹理坐标，不为不同面的顶点共享
5. 纹理映射中查找纹理时的抗锯齿：p260

纹理空间足迹：在纹理中对应于最终栅格化图像方格的区域，求解的话需要先逆投影，再纹理坐标映射，要想做到抗锯齿，就要对纹理空间足迹进行滤波，但是直接计算运算量过大

1. Mipmapping：p265

为了解决纹理空间足迹滤波的计算量过大的问题，引入了多级纹理（Mipmap）进行插值，具体如下：



1. 纹理映射的应用：p267
2. 改变表面颜色：贴花、增加装饰、打印文字、模拟材质
3. 改变其他渲染参数：镜面反射度、镜面粗糙度
4. 法线贴图与凹凸贴图：实际描述的是一个东西，但是法线贴图是凹凸贴图的微分
5. 置换贴图：用一张贴图来移动顶点的位置，通常用大量的小三角形结合置换贴图来模拟地形
6. 阴影贴图：判断一个点是否为阴影：从点光源对场景做透视投影，用z-buffer来判断阴影点与否，进而转化为贴图，并在摄像机中渲染阴影时使用
7. 环境贴图：最直接的是在光线追迹时，光线没有击中任何物体；进一步应用是保存所有的环境光照（光线追迹里面的蒙特卡洛积分或者光栅化过程中的多光源多阴影计算）。基线是球坐标映射，更效率的是立方映射
8. 程序纹理：p273

程序纹理：通过数学方法计算生成，如斑纹纹理、柏林噪声、湍流扰动

# 数据结构

# 光线追迹的深化

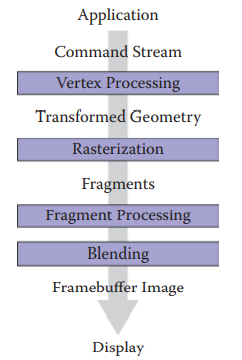
# 采样

# 曲线

# 计算机动画

# 可编程图形硬件的使用

1. 图形硬件管线图：p439



1. CPU与GPU：称为host与device的关系，需要建立映射 p440
2. OpenGL编程（不记录代码细节）：p441
3. 图形硬件编程中三个概念：

缓冲：内存中一种线性结构的数据，将给GPU使用

状态：可理解为图形管线中的某一个阶段

着色器：在GPU上发生的逐点和逐片元的计算

1. 双缓冲技术：p442

有一个背景缓冲和显示缓冲，在背景缓冲中进行计算，在一帧的末尾将背景缓冲与显示缓冲交换，在一帧开始的时候清理背景缓冲

1. 状态机：p443

一些功能通过状态机开启或者关闭，如要开启深度测试等。

Opengl的着色器可参考：<https://www.jianshu.com/p/bdcf7f4a9dcb>

1. 小结vs和fs：p449

顶点着色器：

1. 功能：对顶点进行变换、计算顶点的颜色、处理纹理坐标
2. 输入：来自CPU的attribute：顶点及其附属；来自CPU的全局uniform：变换矩阵、光照参数、颜色等
3. 输出：输出给fs的varying：如法线、该光照方向等，必须经fs处理；内建变量：gl\_Position（必需）\gl\_FrontFacing\gl\_PointSize

片元着色器：

1. 功能：通过光照值、透明度、纹理贴图等来计算每个像素的颜色及其属性，也可以改变像素深度；
2. 输入：来自vs的varying；经vs处理后的内建变量gl\_Position\gl\_FrontFacing\gl\_PointSize，fs的输入都会在光栅化阶段被插值后，再传进来
3. 输出：内建变量：gl\_FragColor
4. VBO与VAO：

顶点缓冲对象：在host里面保存许多顶点及其附属

顶点序列对象：保存VBO中顶点的顺序