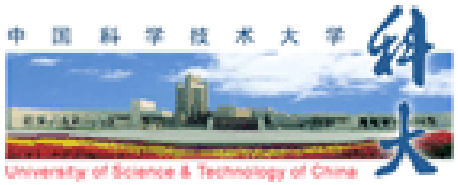
计算机图形学

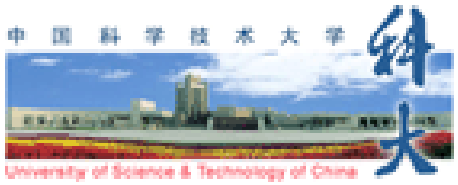
中国科学技术大学数学系
邓建松
dengjs@ustc.edu.cn



第十一章之第三节

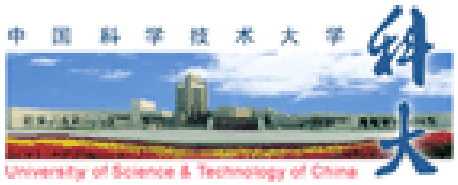
可编程流水线

目标



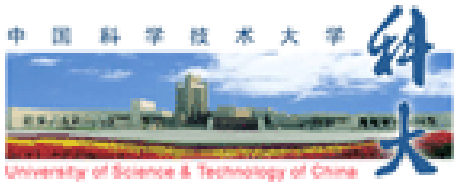
- 介绍可编程流水线
  - 顶点着色器（vertex shader）
  - 片段着色器（fragment shader）
- 介绍着色语言
  - 描述着色器的准备知识
  - RenderMan

介绍



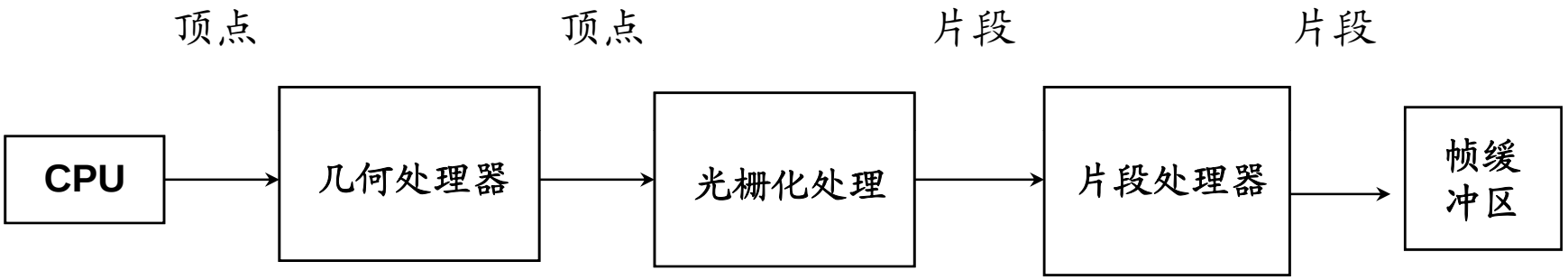
- 近几年在实时图形学方面的主要进展是可编程流水线（programmable pipeline）
  - 由NVIDIA GForce 3首先引入
  - 高端商业显卡支持
    - NVIDIA, ATI, 3D Labs
  - 软件支持
    - Direct X 8 , 9, 10
    - OpenGL Extensions
    - OpenGL Shading Language (GLSL)
    - Cg

背景知识

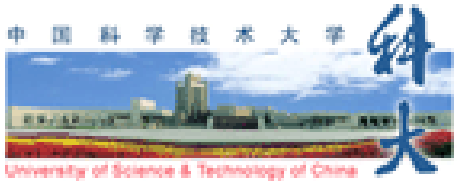


- 由两个成分组成
  - 顶点程序 (着色器)
  - 片段程序 (着色器)
- 需要对两个看起来矛盾过程的深入理解
  - OpenGL流水线
    - 实时
  - RenderMan的想法
    - 离线

黑盒子观点

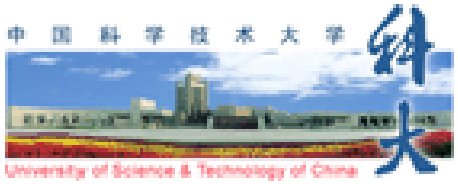


几何计算



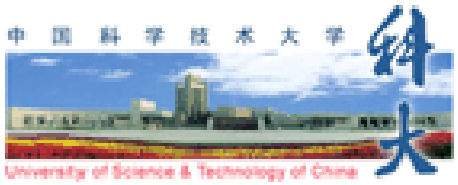
- 几何数据: 顶点集合 + 类型
  - 可以来自于程序、求值器或者显示列表
  - 类型：点、线段、多边形
  - 顶点数据可以是
    - 用顶点的(x,y,z,w)坐标指定(glVertex)
    - 法向量 或 纹理坐标
    - RGBA 颜色
    - 其它数据: 颜色索引、边标志
    - 其它在GLSL中用户定义的数据

逐顶点操作



- 顶点位置由模型视图矩阵变换到视点坐标
- 法向量相应处理
  - 可能需要重新单位化
- 如果激活了纹理自动生成功能，纹理坐标被生成，并且应用可能的纹理矩阵

光照计算

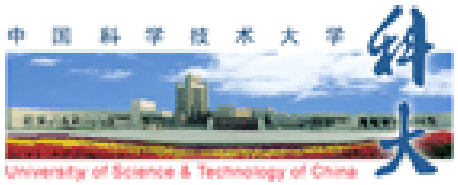


- 进行如下的逐顶点基础上的 Phong光照模型

$$I = k_d I_d \, \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$$

- Phong模型需要在每个顶点处计算 **r**和**v**

OpenGL的光照



- 对Phong模型进行了修改
  - 采用中分（Halfway）向量
  - 全局环境光项
- 细节的标准中有详细定义
- 硬件支持

中分向量

Blinn建议用**n·h**代替 **v·r**，其中

$\mathbf{h} = (\mathbf{l}+\mathbf{v})/|\mathbf{l} + \mathbf{v}|$

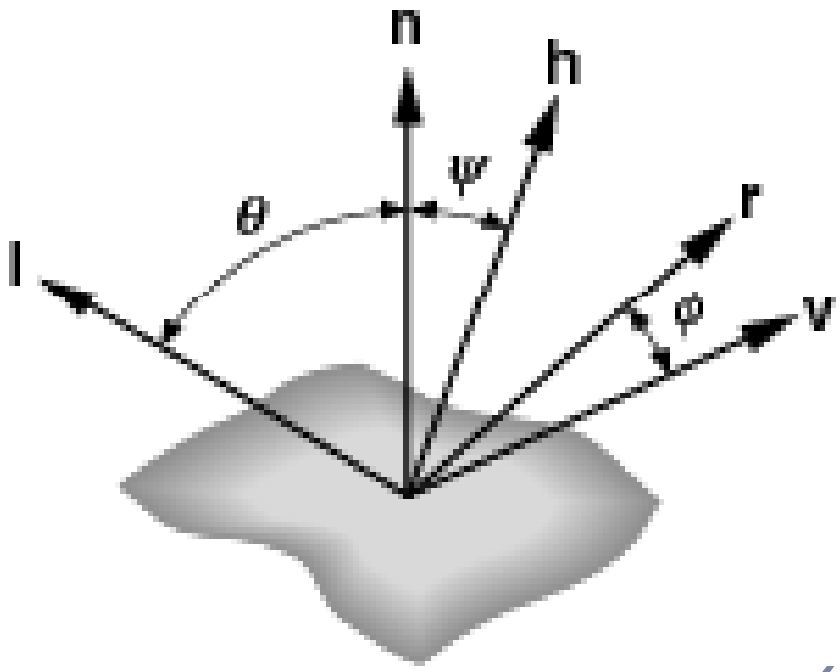
(l+v)/2 是**l**和**v**的平分线

如果**n, l, v** 共面，

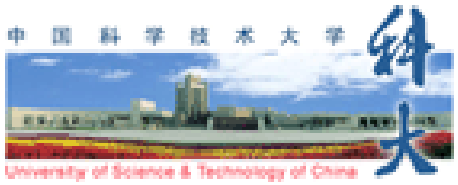
$\psi = \phi/2$

那么就必须调整指数，

使得$(\mathbf{n}\cdot\mathbf{h})^{e'} \approx (\mathbf{r}.\mathbf{v})^{e}$

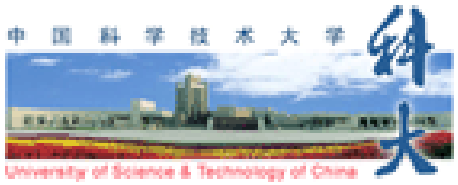


基本单位的集成



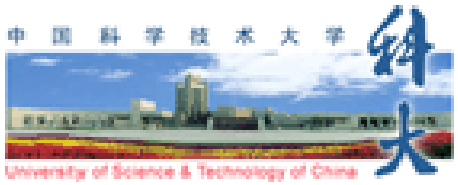
- 顶点接下来被集成为对象
  - 多边形
  - 线段
  - 点
- 用投影矩阵用来变换
- 裁剪
  - 相对于用户定义的平面
  - 视景体$x= \pm w$, $y= \pm w$, $z= \pm w$
  - 裁剪会产生新的顶点
- 透视除法
- 视窗映射

光栅化



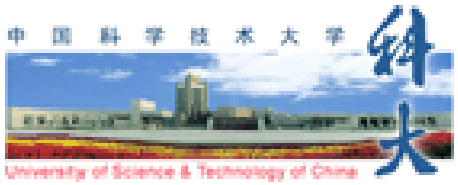
- 几何对象被光栅化成片段（**fragment**）
- 每个片段对应着整数格点的一点：显示出来的像素
- 因此每个片段就是一个潜在的像素
- 每个片段具有
  - 颜色
  - 可能的深度值
  - 纹理坐标

片段操作



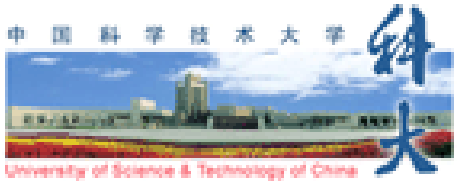
- 纹理生成
- 雾化
- 反走样
- Alpha测试
- 融合
- 逻辑操作
- 模板
- ……

顶点处理器



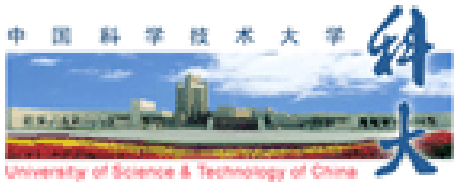
- 取顶点处的下述信息:
  - 位置属性
  - 可能的颜色
  - OpenGL状态
- 输出
  - 在裁剪坐标中的位置
  - 顶点颜色

片段处理器



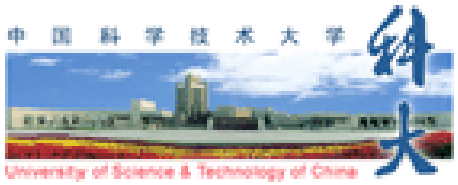
- 从光栅化操作的输出（片段）中得到输入
  - 已由光栅程序把顶点处的值插值得到像素上的值
- 输入一个片段的
  - 颜色
  - 纹理
- 片段还要经过片段测试
  - 隐藏面消除
  - Alpha测试

可编程着色器



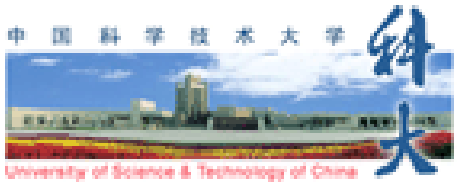
- 把顶点和片段处理中那些固定的功能用可编程处理器代替
- 可能取代两者之一或者同时取代
- 如果采用了可编程程着色器，那么就必须做固定功能处理器的所有事情，或者对其进行了相应的修改

发展

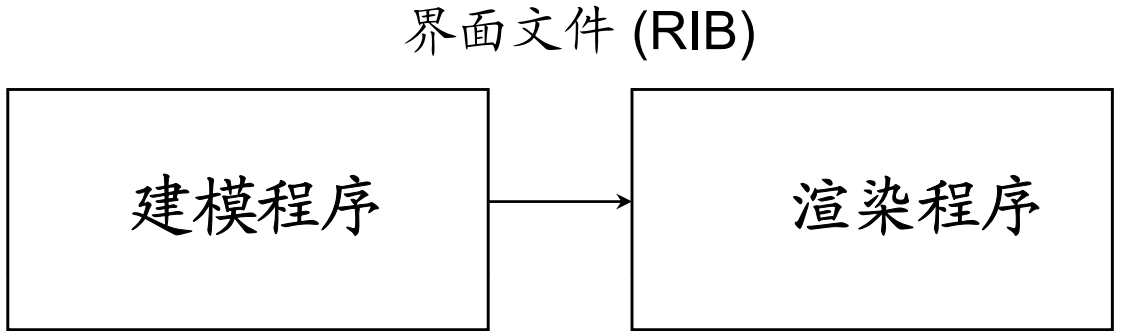


- RenderMan的着色语言
  - 离线渲染
- 硬件着色语言
  - UNC, Stanford
  - NVIDIA
  - OpenGL Vertex Program Extension
  - OpenGL Shading Language
  - Cg --- OpenGL and Microsoft HLSL

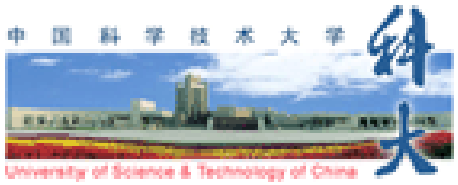
RenderMan



- 由Pixar发展而来
  - 书籍：S. Upstill, *The RenderMan Companion*, Addison-Wesley, 1989.
- 模型

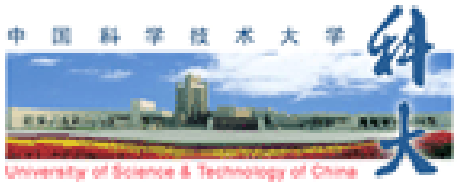


建模 vs 渲染



- 建模程序输出几何模型以及供渲染程序使用的信息
  - 照相机的指定
  - 材料
  - 光照
- 可以用不同类型的渲染程序
  - 光线跟踪
  - 辐射度方法
- 如何定义着色器呢?

着色树

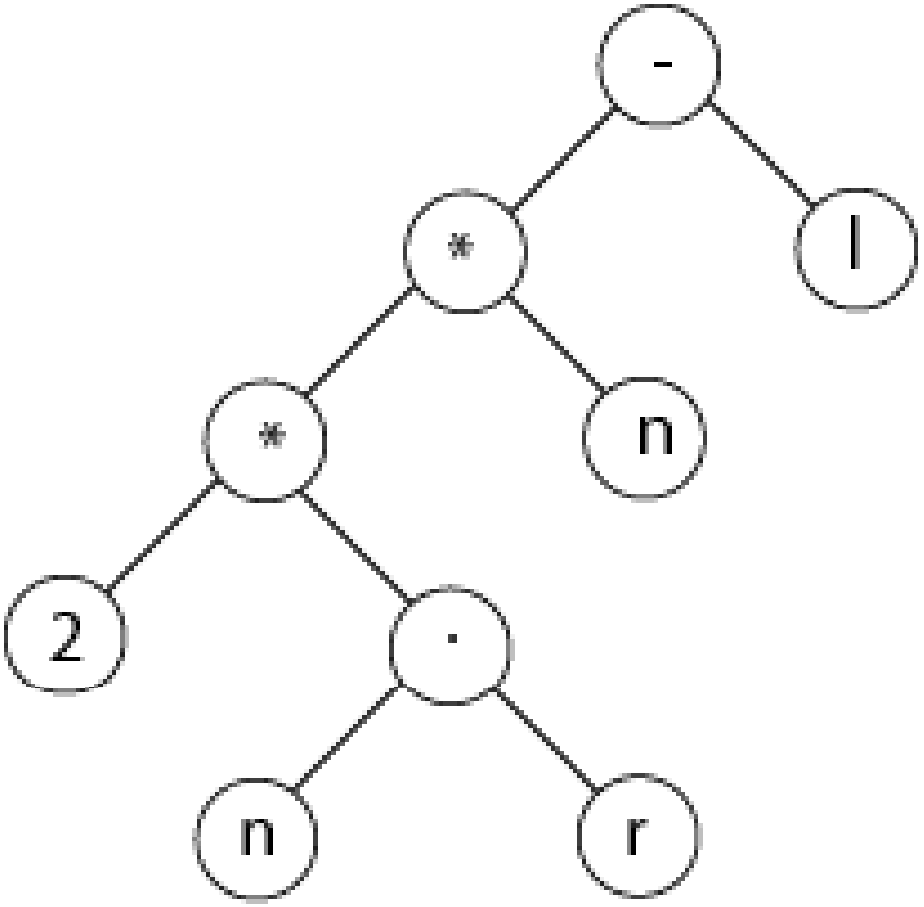


- 基于Phong模型的着色器可以用下述代数表达式表示:

$$I = k_d I_d \, \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^s + k_a I_a$$

- 这个表达式可以用树形结构描述
- 需要类似于点积和外积等新运算符以及类似于矩阵和向量等新数据类型
- 环境变量是状态的一部分

反射向量



Phong模型