计算机图形学
Computer Graphics

陈中贵
chenzhonggui@xmu.edu.cn
http://graphics.xmu.edu.cn/~zgchen

Graphics@XMU

*课件来源于中科大黄章进副教授讲义

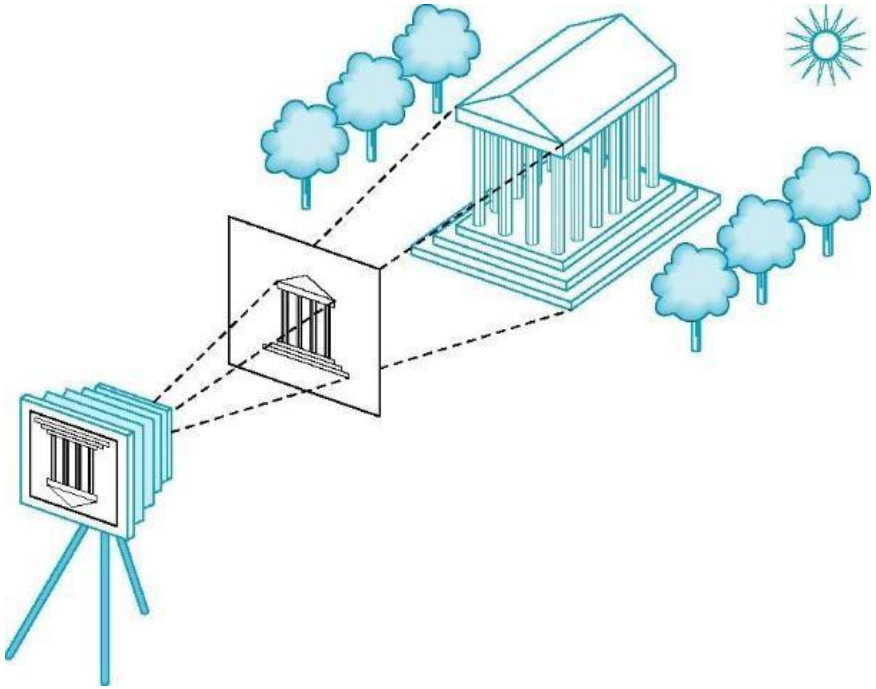
渲染流水体系及编程接口

主要内容

- 图形渲染流水线体系结构
- 应用程序编程接口（API）

成像模型回顾

- 能否模拟虚拟照相机模型设计图形系统中的硬件和软件？
- 应用程序编程接口(API)
  - 只需指定
    - 对象
    - 材料属性
    - 观察者
    - 光源
- 如何实现API？

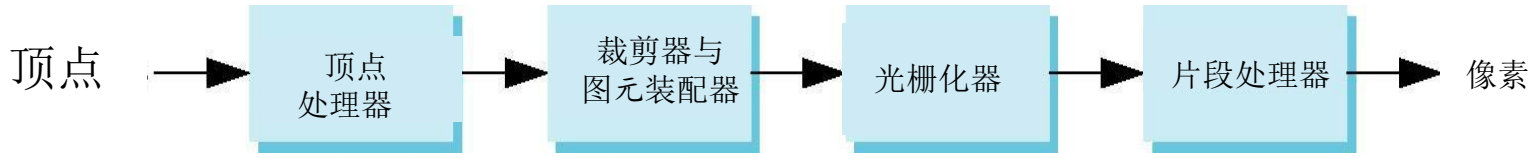


成像模型回顾

- 全局光照模型
  - 光线跟踪方法
  - 辐射度方法
  - 计算量非常大，目前没有硬件加速支持，不适合实时或交互式系统
- 局部光照模型
  - Phong光照模型
  - 计算量适中，有硬件加速支持
  - 当前主流API：OpenGL, Direct3D

流水线体系

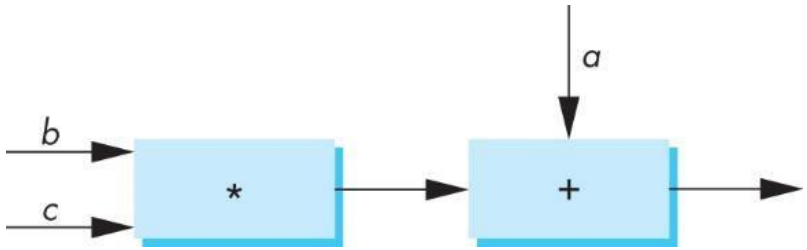
- 按照应用程序定义对象的先后顺序，依次处理每个对象
  - 只考虑局部光照
- 流水线体系结构



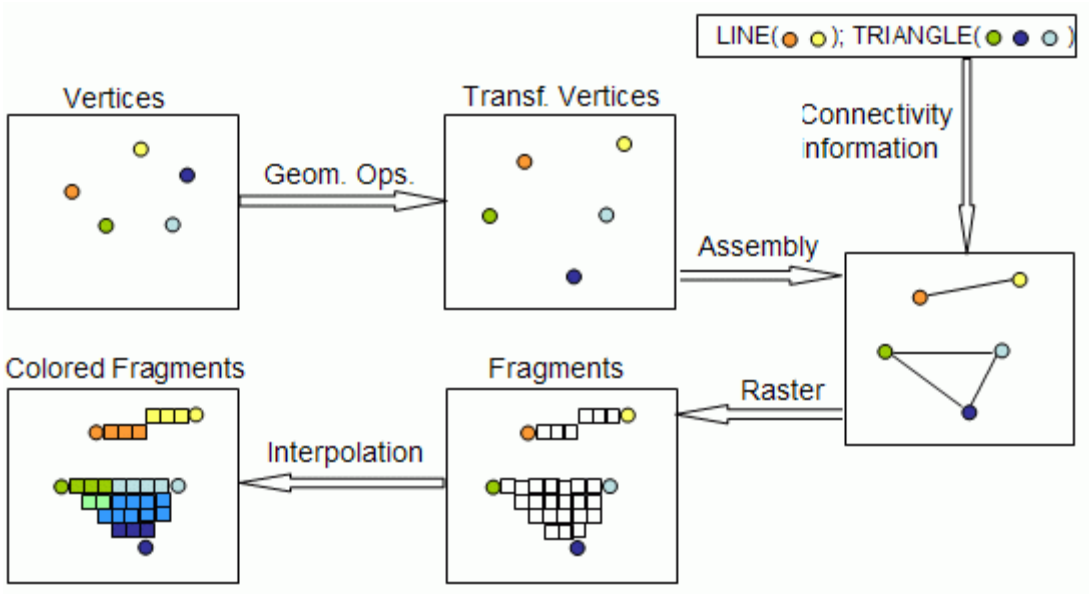
- 所有步骤都可以通过显示卡的硬件实现

流水线(pipeline)

- 工厂装配线
- 算术流水线：a+(b*c)



- 图形渲染流水线

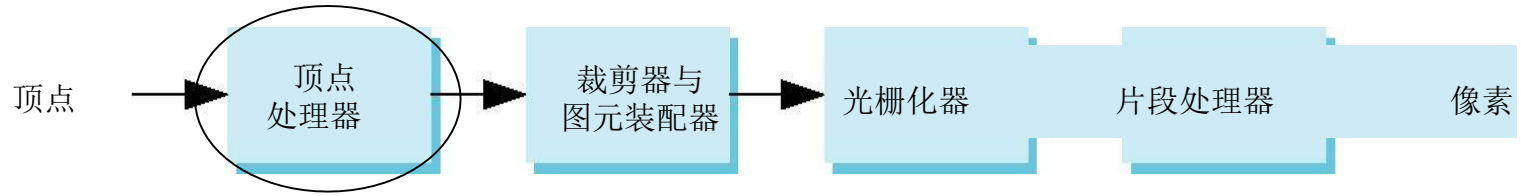


成像过程

- 四个主要步骤：
  - 顶点处理
  - 裁剪和图元装配
  - 光栅化
  - 片段处理

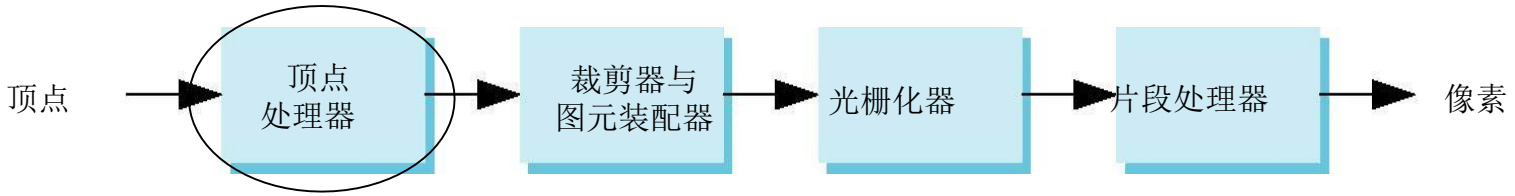
顶点处理

- 图元的类型和顶点集定义了场景的几何
  - 对象由一组图元组成，而每个图元又包含一组顶点
- 流水线中大部分工作是把对象在一个坐标系中表示转化为另一坐标系中的表示：
  - 世界坐标系
  - 照相机(眼睛)坐标系
  - 屏幕坐标系
- 坐标的每个变换相当于一次矩阵乘法
- 顶点处理器也计算顶点的颜色



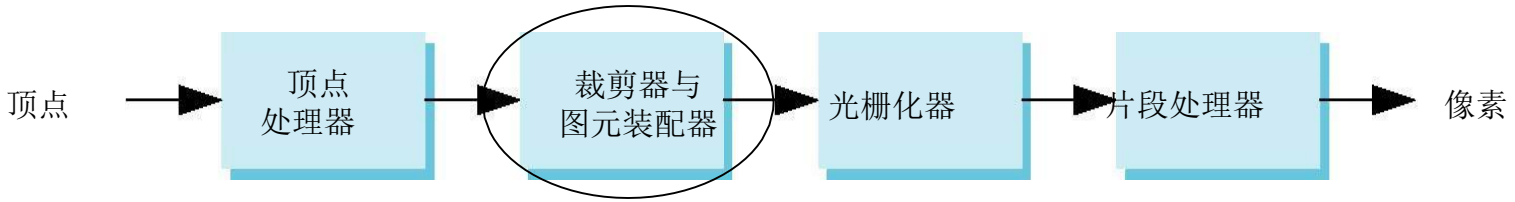
投影

- 把三维观察者位置与三维对象结合在一起，确定二维图像的构成
  - 透视投影：所有投影线交于投影中心
  - 平行投影：投影线平行，投影中心在无穷远，用投影方向表示
- 在顶点处理步中，对各个顶点的处理是相互独立的



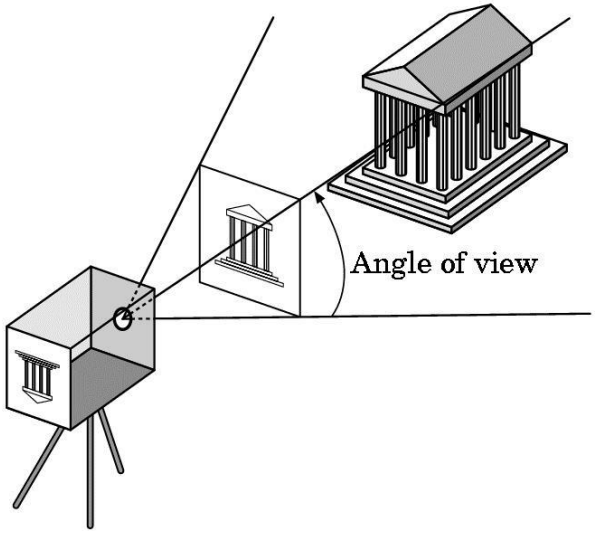
图元装配

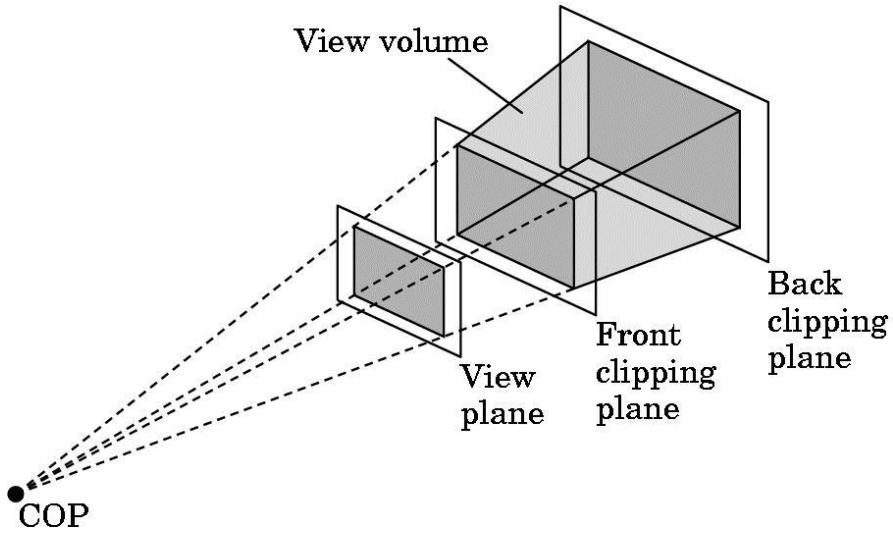
- 在进行裁剪和光栅化处理之前，顶点必须组装成几何对象
  - 线段
  - 多边形
  - 曲线和曲面



裁剪

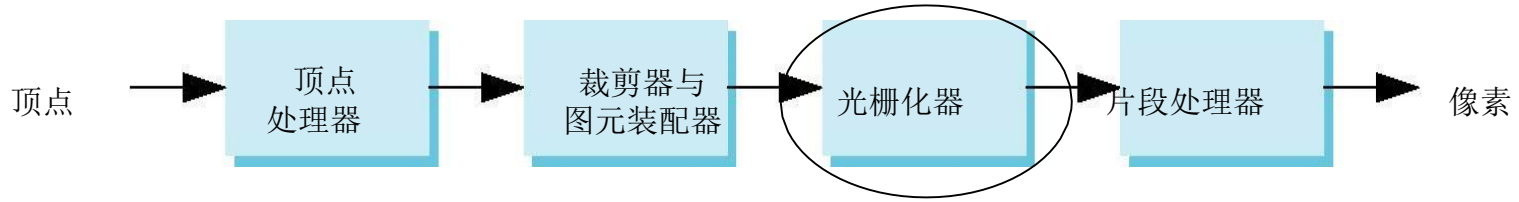
- 真正的照相机不能“看到”整个世界，图形学中的虚拟照相机也只能看到世界的一部分
  - 不在视景体中的对象要从场景中裁剪掉
- 裁剪必须针对逐个图元进行





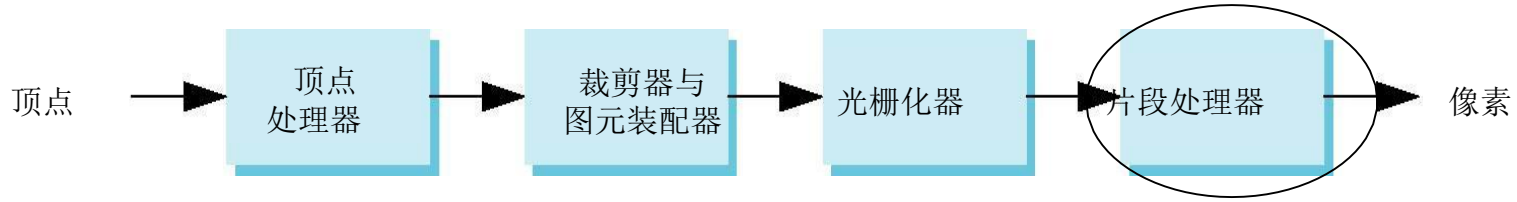
光栅化

- 如果一个对象不被裁掉，那么在帧缓冲区中相应的像素就必须被赋予颜色
- 光栅化程序为每个图元生成一组片段
- 片段是“潜在的像素”
  - 在帧缓冲区中有一个位置
  - 具有颜色和深度属性
- 光栅化程序在对象上对顶点的属性进行插值得到片段的属性

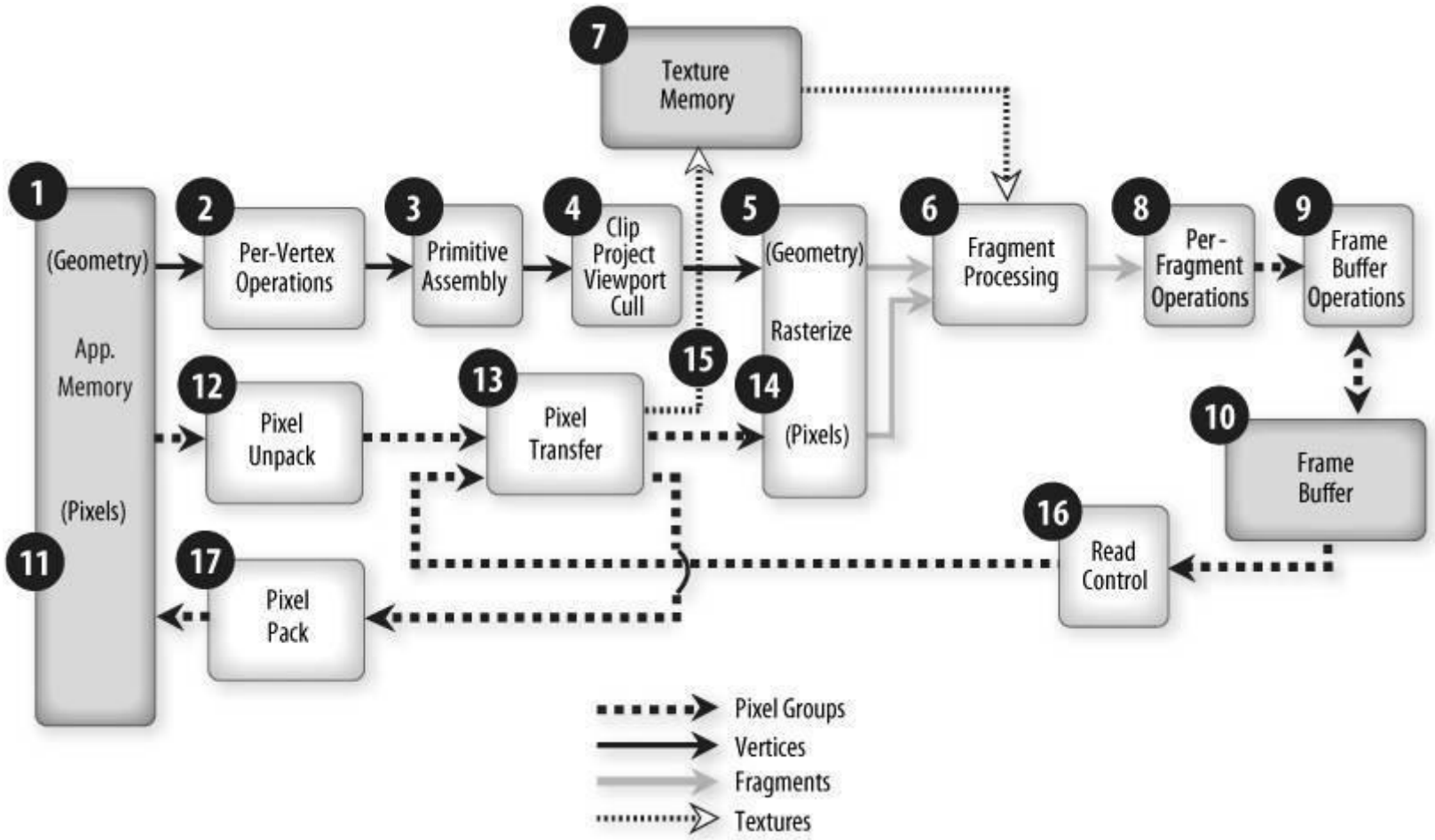


片段处理

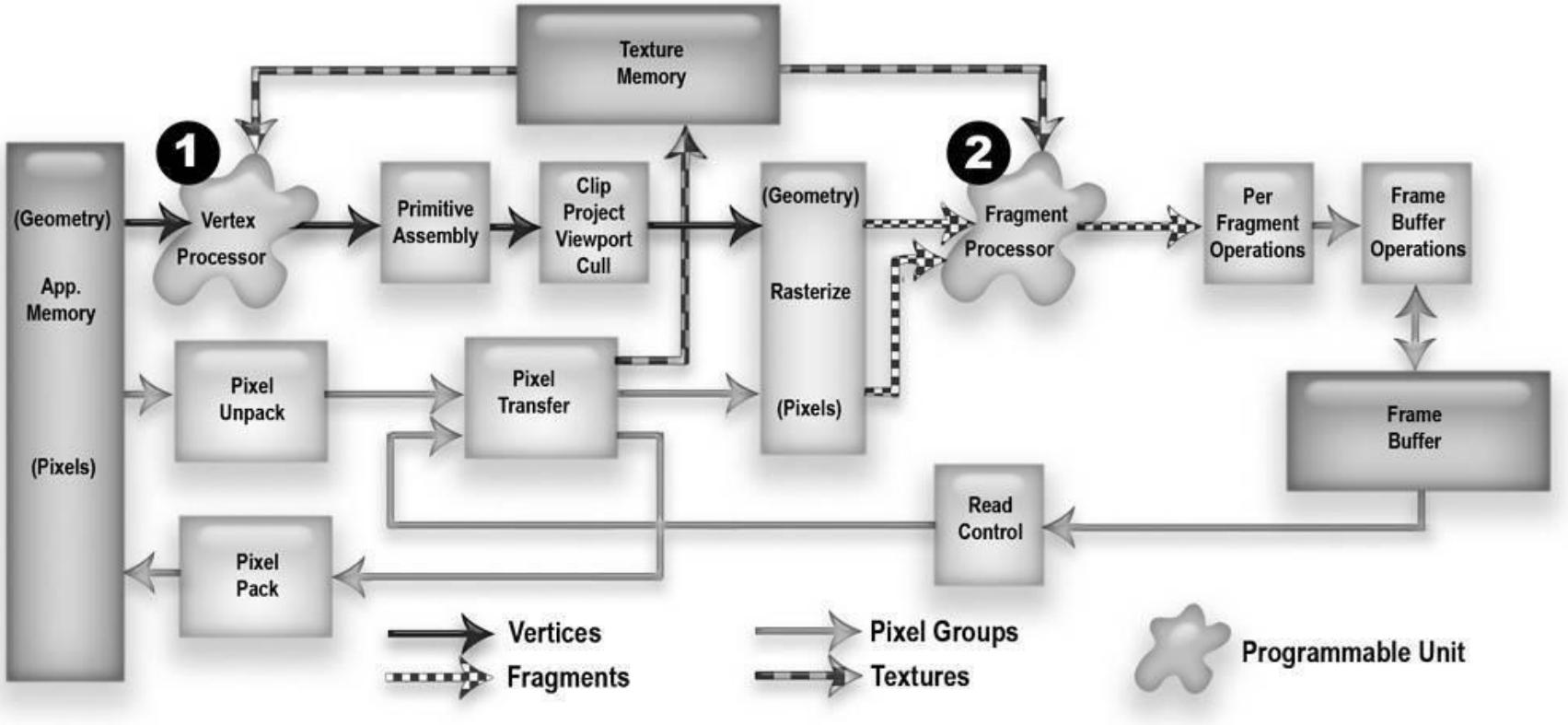
- 对片段进行处理，以确定帧缓冲区中相应像素的颜色
- 颜色可以由纹理映射确定，也可以由顶点颜色插值得到
- 片段可能被离照相机更近的其它片段挡住
  - 隐藏面消除



固定功能流水线

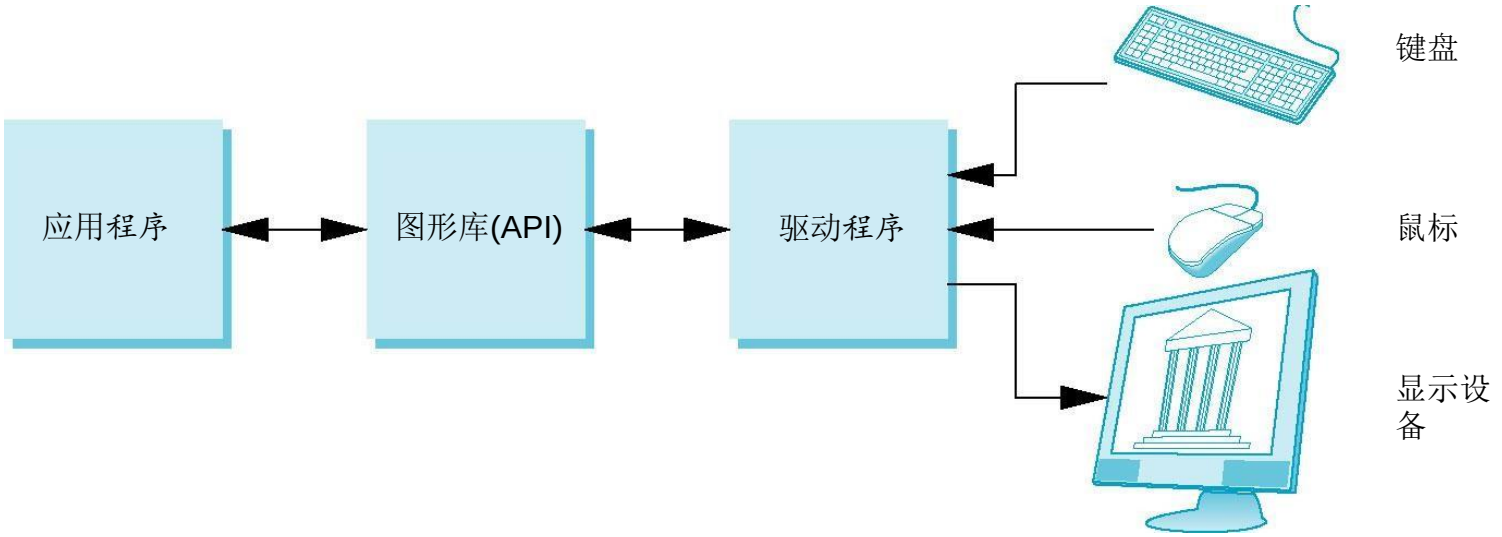

1
(Geometry)
App.
Memory
(Pixels)
2
Per-Vertex
Operations
3
Primitive
Assembly
4
Clip
Project
Viewport
Cull
5
(Geometry)
Rasterize
(Pixels)
6
Fragment
Processing
7
Texture
Memory
8
Per-
Fragment
Operations
9
Frame
Buffer
Operations
10
Frame
Buffer
11
12
Pixel
Unpack
13
Pixel
Transfer
14
15
16
Read
Control
17
Pixel
Pack
Pixel Groups
Vertices
Fragments
Textures

可编程流水线


Texture Memory
1
2
(Geometry)
App. Memory
(Pixels)
Vertex Processor
Primitive Assembly
Clip Project Viewport Cull
(Geometry)
Rasterize
(Pixels)
Fragment Processor
Per Fragment Operations
Frame Buffer Operations
Frame Buffer
Pixel Unpack
Pixel Transfer
Read Control
Pixel Pack
Vertices
Fragments
Pixel Groups
Textures
Programmable Unit

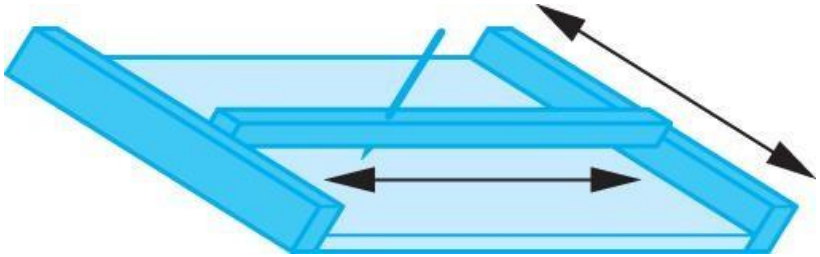
编程接口

- 应用程序设计人员是通过软件接口接触图形系统，这个接口就是应用编程接口(API)



笔式绘图仪模型

- 二维模型
- PostScript的设计基础
- 基本函数：
  - moveto(x,y)：移位
  - lineto(x,y)：画线
  - 初始化/终止
  - 改变画笔颜色或线宽



例子

- 程序片段：

  moveto(0,0)

  lineto(1,0)

  lineto(1,1)

  lineto(0,1)

  lineto(0,0)

(a)

基于光栅的二维系统

- 基于光栅：把像素直接写到帧缓存中
- 单个函数

  write_pixel(x, y, color)

  – (x,y)是帧缓存中像素的位置
  – color是该位置像素的颜色

三维API的构成

- 函数：定义生成一幅图像所需要的内容
  - 对象
  - 观察者
  - 光源
  - 材料属性
- 其它信息
  - 从鼠标和键盘等设备获取输入
  - 系统的能力

对象的定义

- 绝大多数API支持有限的基本几何对象，例如：
  - 点 points（零维对象）
  - 线段 line segments（一维对象）
  - 多边形 polygons（二维对象）
  - 某些曲线和曲面
    - 二次曲面 quadrics
    - 多项式参数曲面
- 所有基本形状都是通过空间中的位置或顶点(vertices)定义的。

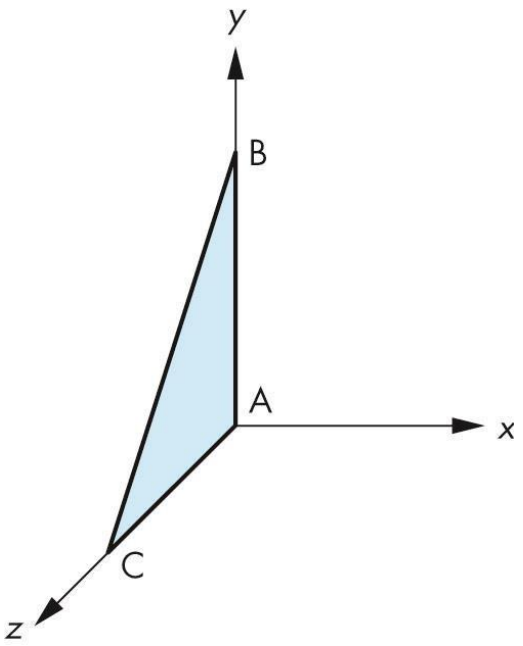
例子（旧式风格）

对象类型

顶点位置

```
glBegin(GL_POLYGON)
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(0.0, 1.0, 0.0);
  glVertex3f(0.0, 0.0, 1.0);
glEnd( );
```

对象定义结束



例子（基于GPU的）

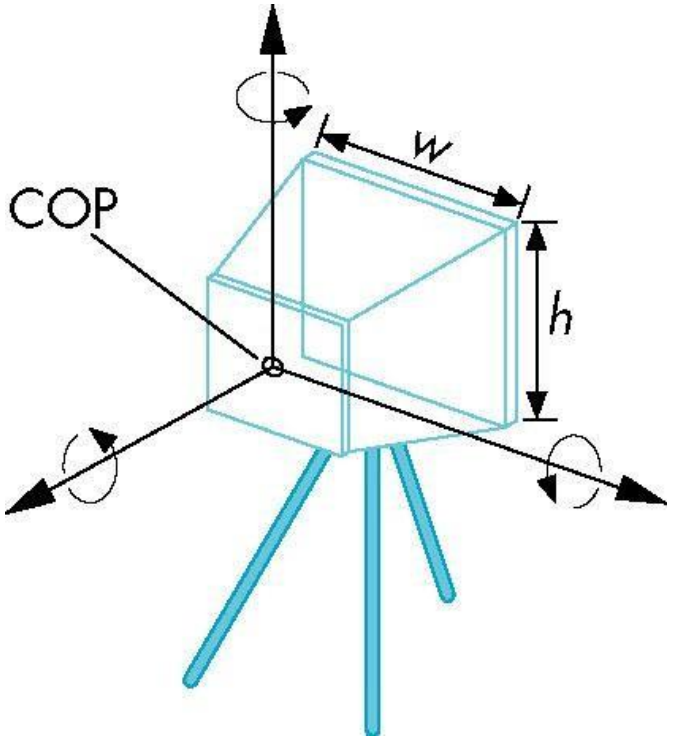
- 把几何数据放在数组中

```
vec3 points[3];
points[0] = vec3(0.0, 0.0, 0.0);
points[1] = vec3(0.0, 1.0, 0.0);
points[2] = vec3(0.0, 0.0, 1.0);
```

- 把数据传送到GPU
- 告诉GPU绘制成三角形

照相机的指定

- 六个自由度
  - 镜头中心的位置，即投影中心(COP)
  - 方向
- 镜头、焦距
- 胶卷尺寸
- 胶卷平面的方向

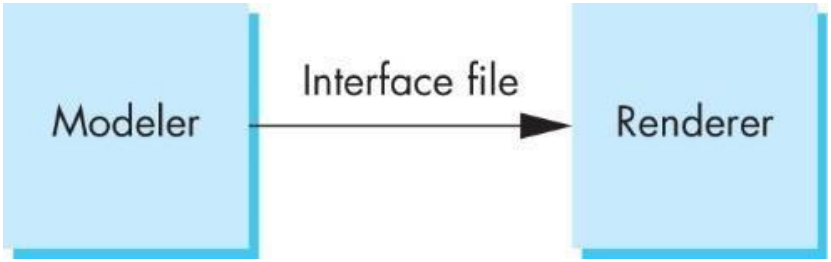


光源与材料

- 光源类型
  - 点光源与分布式光源
  - 聚光灯
  - 远光源与近光源
  - 光源的颜色属性
- 材料属性
  - 吸收性：颜色属性
  - 反射性：漫反射、镜面

建模-渲染模式

- 建模器：交互式地设计并摆放对象
  - 工作站
  - 基于虚拟照相机模型进行实时渲染的建模软件
- 渲染器：生成高质量的图像
  - 渲染农场：计算集群
  - RenderMan、POV-Ray：基于光线跟踪
- 接口文件：描述要渲染的对象以及光源、相机和材料属性等信息的（文本）文件



本章小结

- 计算机图形系统的组成
- 虚拟照相机模型
- 图形处理流水线体系结构
- 图形API的构成