

# 计算机图形学

唐慧

影像科学与技术实验室

东南大学计算机学院

[corinna@seu.edu.cn](mailto:corinna@seu.edu.cn)

## 2：图形系统

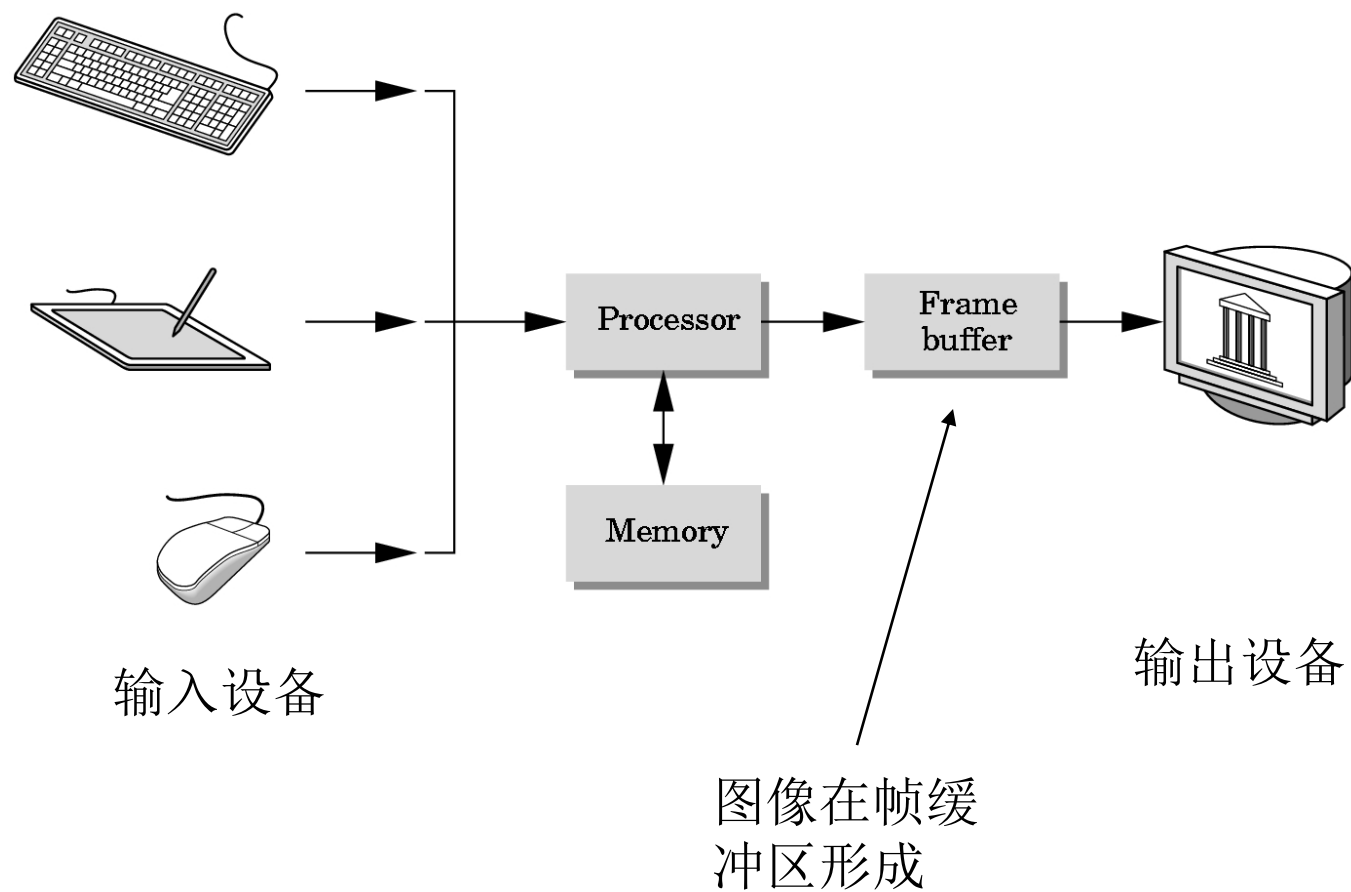
---

- 2.1 基本图形系统
- 2.2 图像的形成
- 2.3 虚拟照相机模型
- 2.4 程序员接口（API）
- 2.5 图形系统的体系结构
- 2.6 可编程流水线



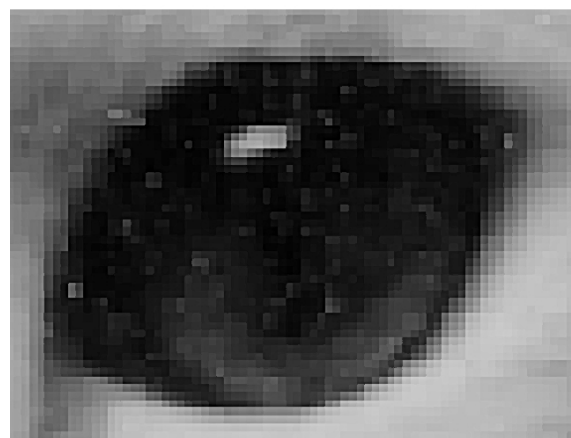
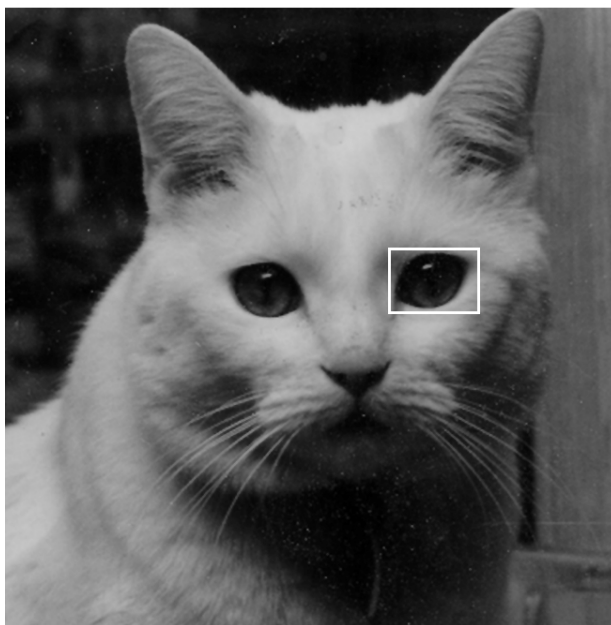
## 2.1 基本图形系统

# 基本图形系统



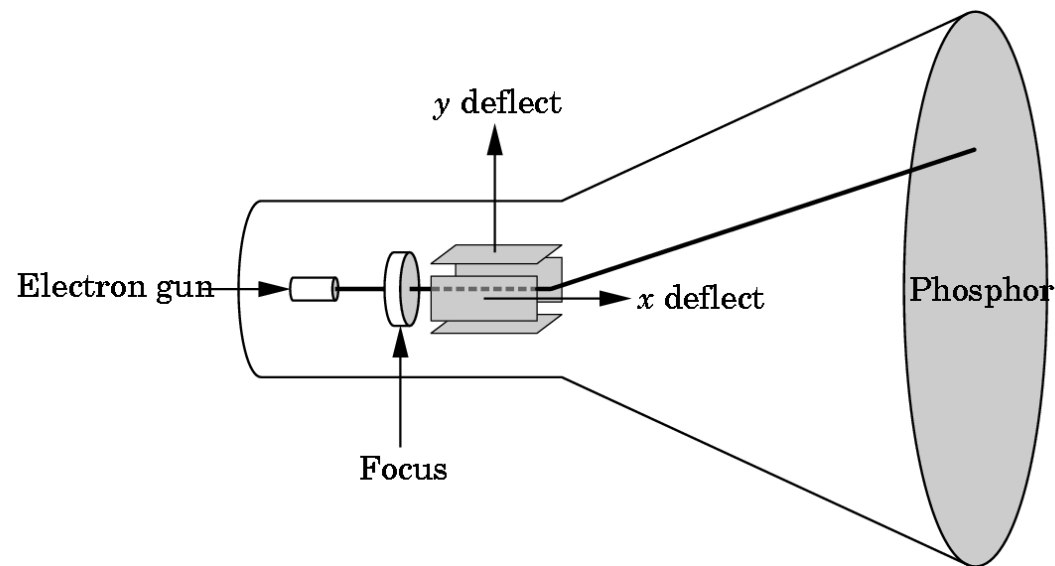
# 像素和帧缓存

- 图像以像素数组的形式存贮在帧缓冲区中



# 输出设备：显示器

- 显示内容为帧缓冲区中存储格式的直观反映
- 典型分辨率有 800x600 1024x768 1280x800 1280x1024



## 思考题

---

- 制作电影的胶片所具有的分辨率大约是  $2000*3000$ , 这样的分辨率对于制作与电影画质相当的电视动画意味着什么?



## 2.2 图像的形成（成像）





# 本节内容

---

- 成像的基本概念
- 成像的物理基础
  - 光照
  - 颜色
  - 感知



# 成像：物理的和合成的

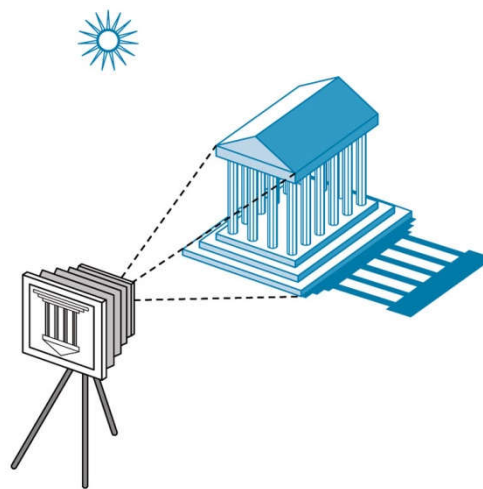
---

- 计算机图形学中，图像的构造过程类似于物理方法成像的过程
  - 照相机
  - 显微镜
  - 望远镜
  - 人类视觉系统



# 图像构造的要素

- 对象
- 观察者
- 光源

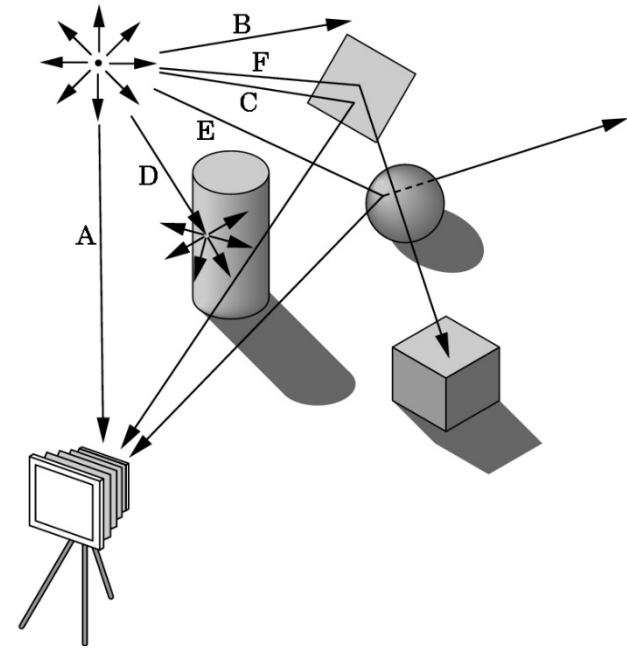


- 材料的属性确定光照射在物体上的效果
- 注意物体对象、观察者以及光源三者完全独立。

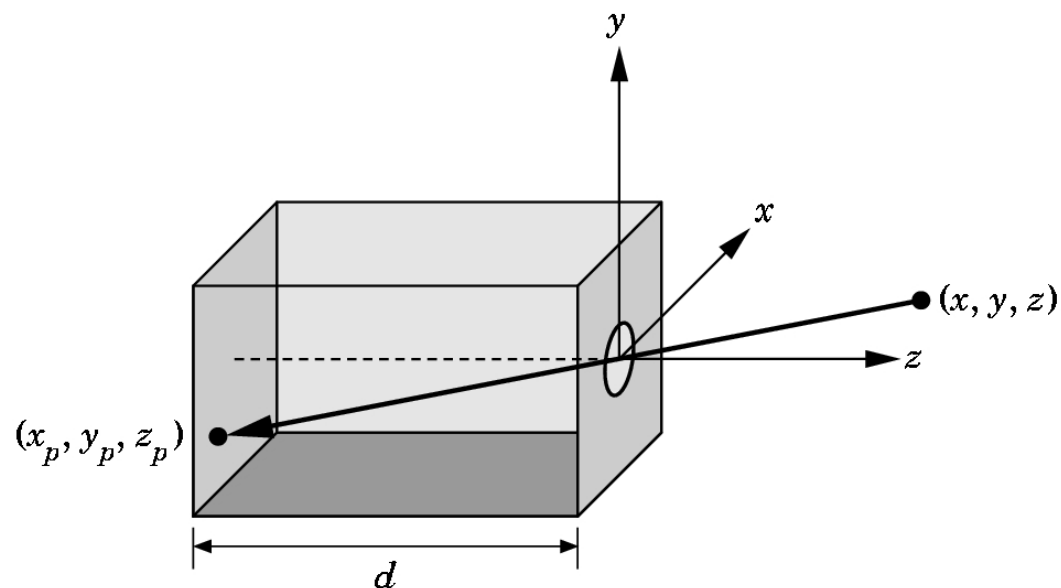


# 成像物理模型

- 从一个点光源出发，跟踪所有的光线，确定哪些光线进入照相机的镜头
- 每条光线在被吸收或者进入无穷空间之前，有可能与物体发生多次接触。



# 成像系统：针孔相机



$(x, y, z)$ 点的投影位置

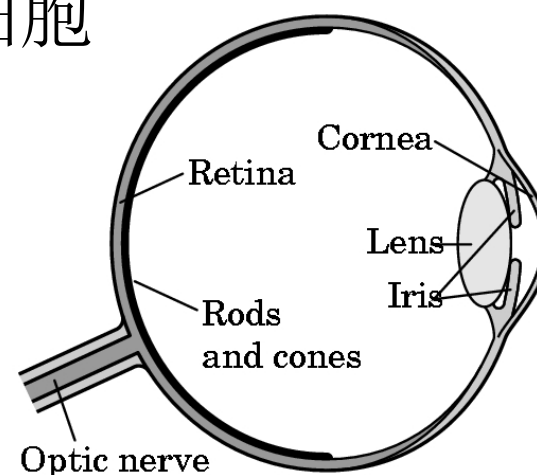
$$x_p = -x/z/d \quad y_p = -y/z/d \quad z_p = -d$$



# 成像系统：人类视觉系统

- 人类视觉系统有两种类型的感光细胞

- 杆状细胞 Rods: 单色的、夜视
- 锥状细胞 Cones
  - 感受彩色
  - 三种类型
  - 只有三种感受值（刺激）送到大脑



# 三色理论

---

- 只需匹配这三种值，即所谓的三原色(three primary colors)
  - R(ed)、G(reen)、B(lue)
  - 在计算机图形学中所谓的三原色不一定与人眼所感受的三种值恰好完全匹配



# 加色与减色

## ○ 加色

- 把三原色的值加在一起形成一种颜色
  - CRT, 投影仪, 正片
- Red (R), Green (G), Blue (B)

## ○ 减色

- 通过在白光中过滤掉蓝绿色(cyan), 洋红色(magenta)和黄色(yellow)形成最终的颜色
  - 光线和物体的相互作用
  - 彩色打印
  - 负片



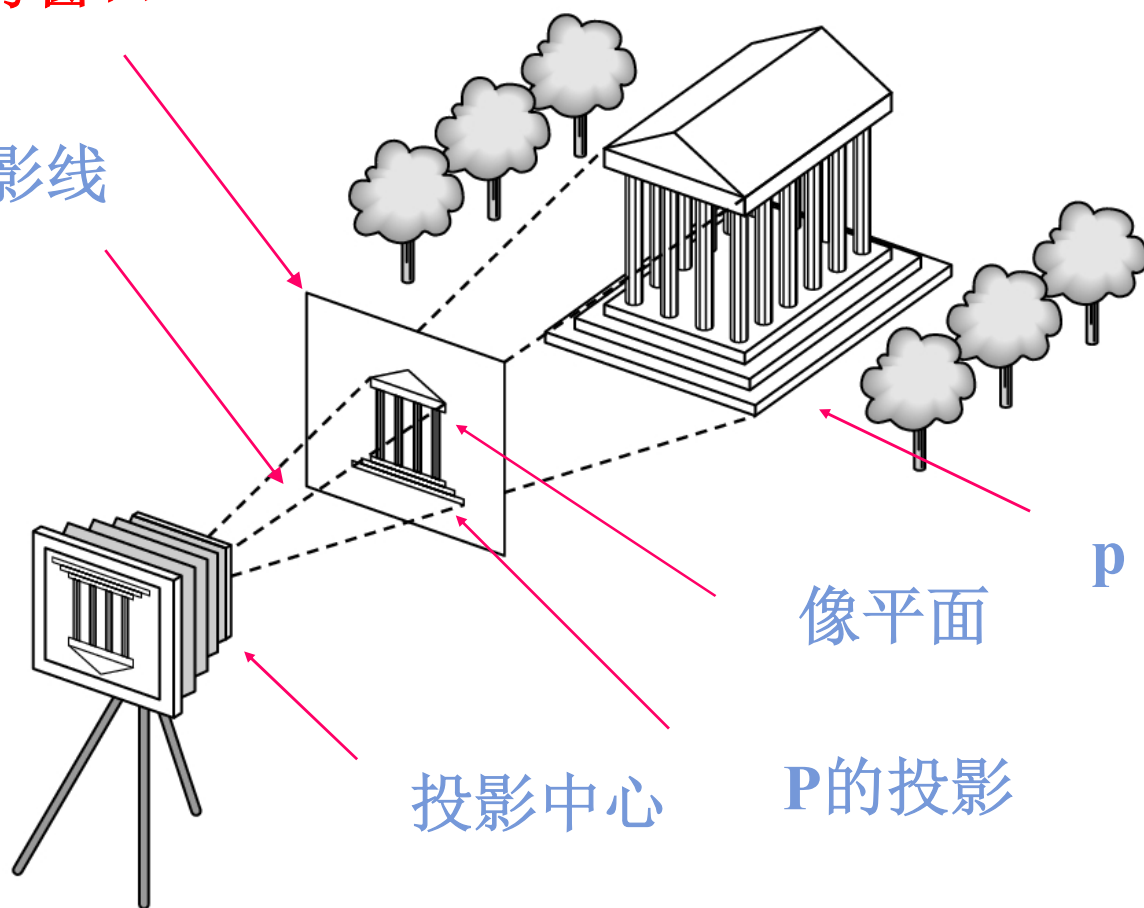


## 2.3 虚拟照相机模型

# 虚拟照相机模型

裁剪窗口

投影线



# 虚拟相机模型优点

---

- 把对象、观察者和光源区分开
- 二维图形是三维图形的一个特殊情形
- 可以得到简单的应用程序接口API (Application programming interface)
  - 指定对象、光源、照相机、材质属性
  - 由API的实现确定最终的图像
- 可以得到快速的硬件实现。



# 回顾图像形成

---

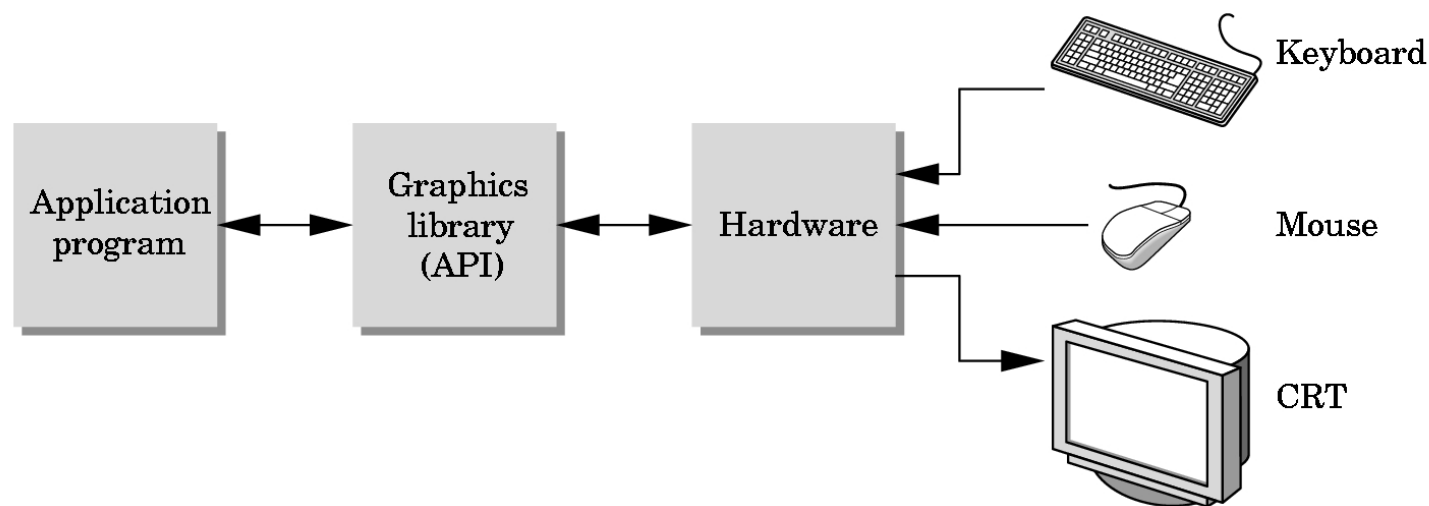
- 能否模拟虚拟照相机模型来设计图形系统中的软硬件？
- 应用程序接口 (API)
  - 用户只需指定
    - 对象
    - 材料
    - 观察者
    - 光源
- 如何实现API？



## 2.4 程序员接口

# 应用程序接口

- 程序设计人员是通过接口与图形系统打交道，这个接口就是应用程序接口（API）



# API 构成

---

- 通过函数定义构成一幅图像所需要的内容
  - 对象
  - 观察者
  - 光源
  - 材质属性
- 其它信息
  - 从鼠标和键盘等设备获取的输入信息
  - 系统配置



# 对象定义

---

- 场景对象由基本图元定义
- 绝大多数API支持一些基本图元，例如：
  - 点 Points (1D object)
  - 线段 Line segments (2D objects)
  - 多边形 Polygons (3D objects)
  - 某些曲线和曲面
    - 二次曲面 Quadrics
    - 多项式参数曲面
- 所有基本图元都是通过空间中的位置或者说顶点(vertices)定义的。





## 示例（旧版）

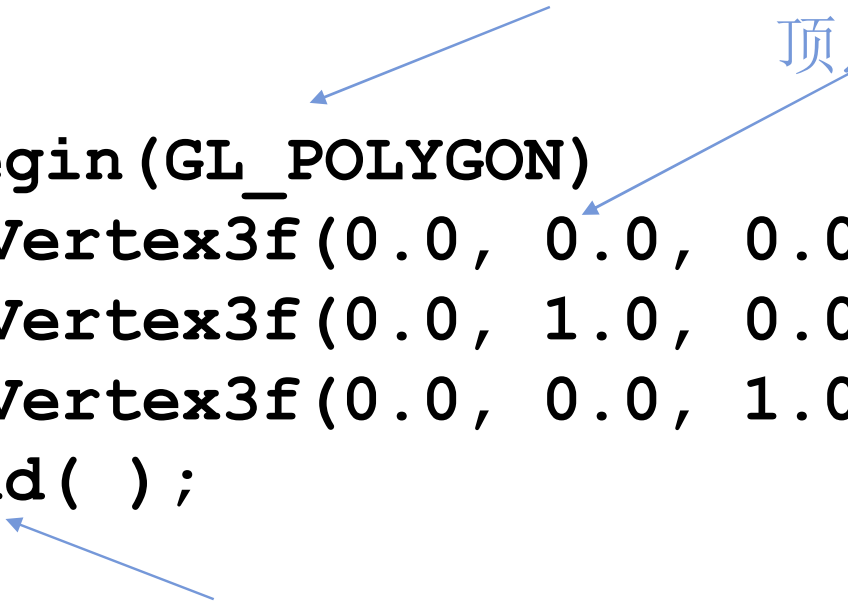
---

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



# 示例 (基于GPU)

---

- 将几何数据置于一数组中

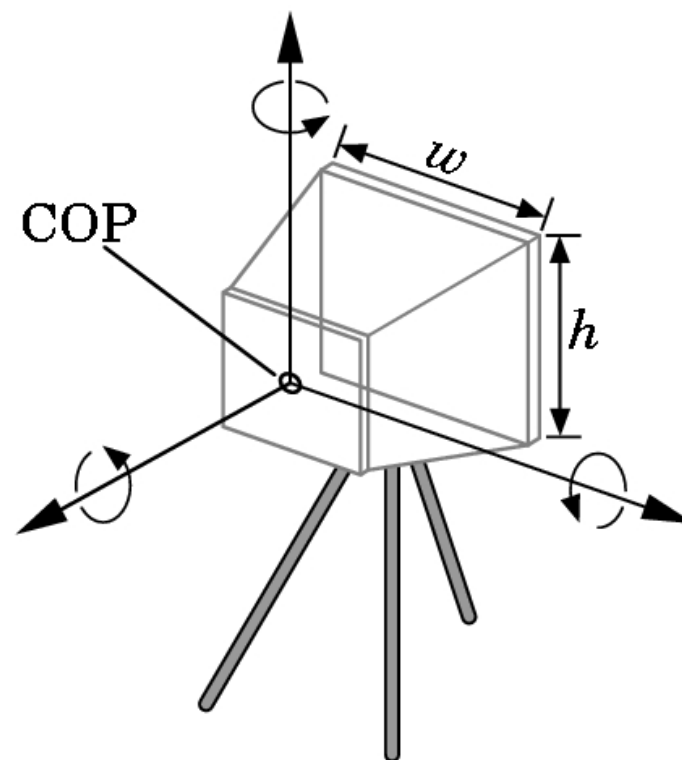
```
var points = [  
    vec3(0.0, 0.0, 0.0),  
    vec3(0.0, 1.0, 0.0),  
    vec3(0.0, 0.0, 1.0),  
];
```

- 将数组发送到GPU
- 告诉 GPU 绘制一个三角形



# 照相机的指定

- 位置：投影中心（COP）
- 方向：相机坐标系
- 焦距
  - 决定了在胶卷平面上成像尺寸
- 胶卷平面
  - 大小
  - 方向



# 光源与材质

---

- 光源类型
  - 点光源与分布式光源
  - 聚光灯
  - 光源的远近
  - 颜色属性
- 材质属性
  - 吸收性：颜色属性
  - 反射性
    - 漫反射
    - 镜面反射

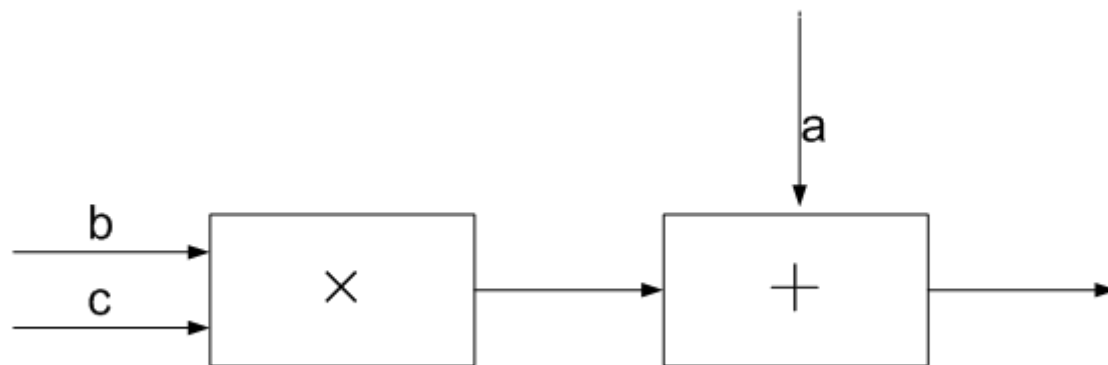


## 2.5 图形系统的体系结构



# 流水线体系结构

- 流水线概念
- 算术流水线



- 图形绘制流水线



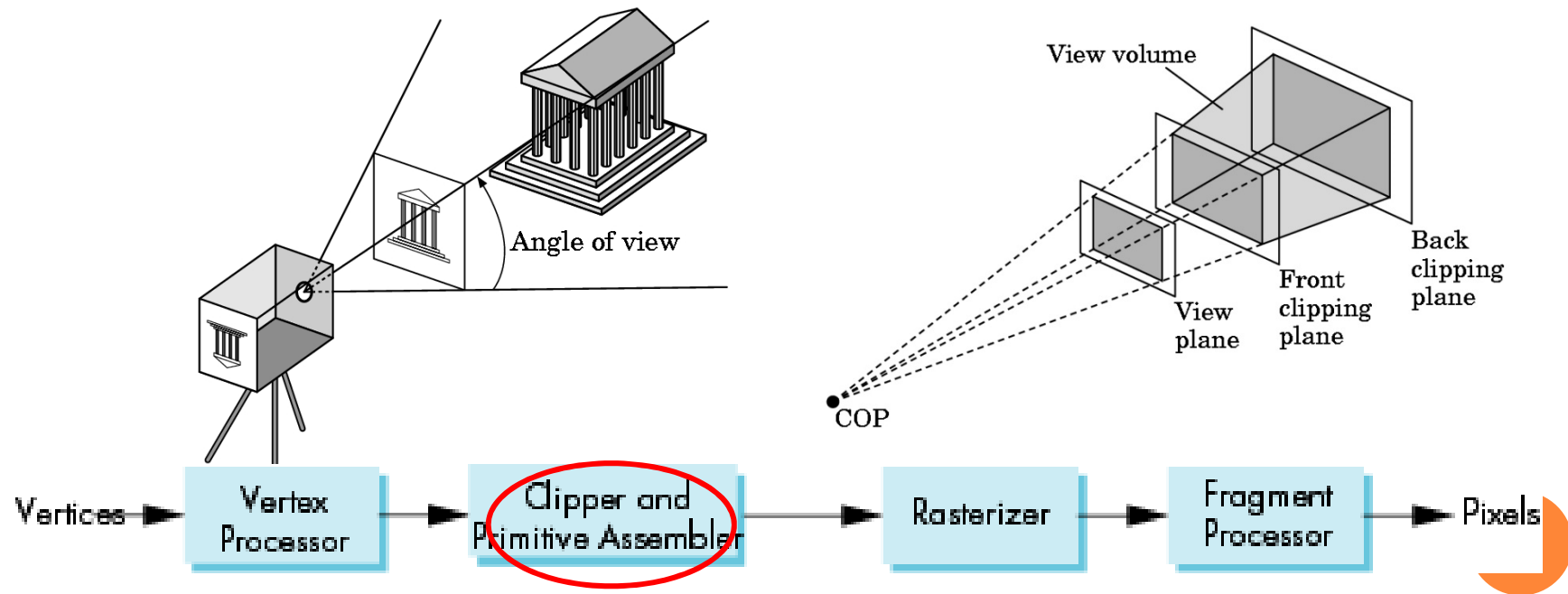
# 顶点处理

- 流水线中大部分工作是把对象从一个坐标系中变换到另一坐标系：
  - 世界坐标系
  - 相机坐标系
  - 屏幕坐标系
- 每个变换相当于一次矩阵乘法
- 顶点处理器还计算顶点颜色



# 裁剪

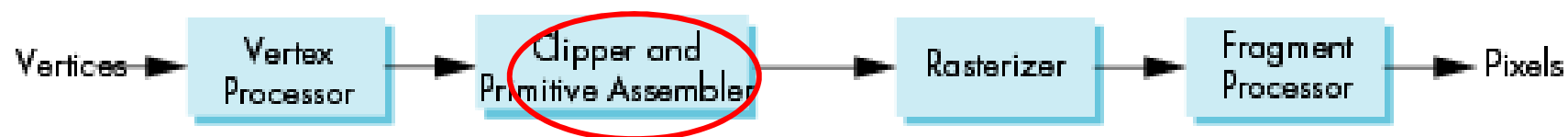
- 真正的照相机不能“看到”整个世界，图形学中虚拟的照相机也只能看到世界的一部分
  - 不在下述视景体中的对象要从场景中裁剪掉





# 图元组装

- 裁剪过程必须对图元逐个进行，而不是对顶点。因此在进行裁剪之前，必须把顶点组合成图元
  - 线段
  - 多边形
  - 曲面



# 光栅化

- 如果一个对象在图像中可见，那么就需要确定帧缓冲区相应像素的颜色
- 光栅化为每个物体确定一系列的片元(fragment)
- 片元是“潜在像素”
  - 对应于帧缓存中的像素
  - 有颜色和深度信息
- 光栅化过程把顶点属性进行插值处理



# 片元处理

- 片元处理决定帧缓存中对应像素的颜色
- 颜色会受到纹理或者顶点颜色插值的影响
- 由于更靠近相机的片元遮挡，某些片元可能不可见
- 不完全遮挡 → 半透明效果



## 2.6 可编程流水线

# 可编程流水线

- 可编程部分
  - 顶点处理模块
  - 片元处理模块

